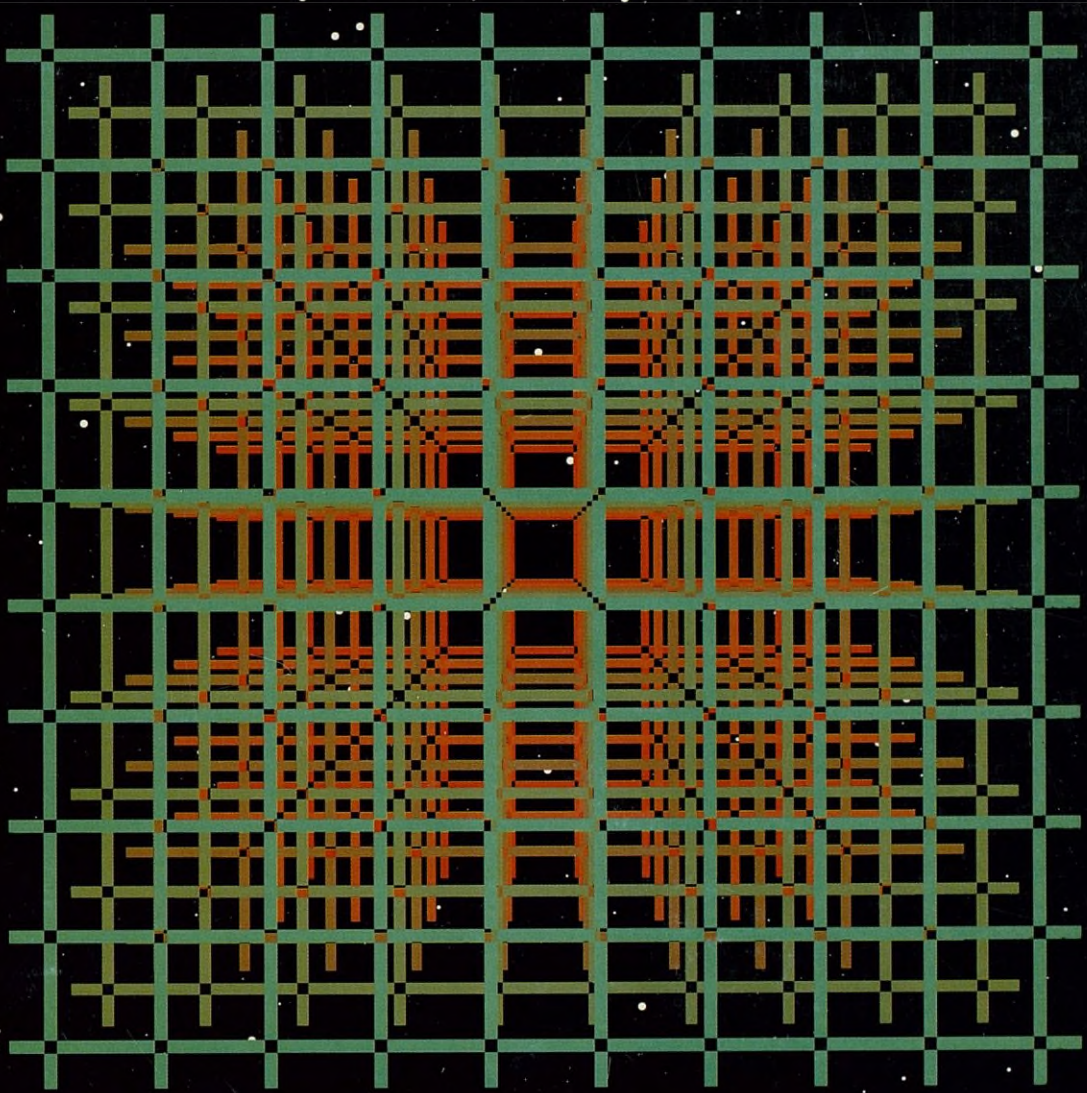
FUNDAMENTOS DIGITALES

JAVIER HOLGADO CORRALES



SERVICIO DE PUBLICACIONES DE LA UNIVERSIDAD DE CADIZ

FUNDAMENTOS DIGITALES

JAVIER HOLGADO CORRALES

Ingeniero Industrial.
Profesor Titular de Circuitos y Sistemas Digitales
de la Universidad de Cádiz.

SERVICIO DE PUBLICACIONES DE LA UNIVERSIDAD DE CADIZ

Servicio de Publicaciones de la Universidad de Cádiz
Dpto. Legal: CA-801-92
I.S.B.N.: 84-7786-103-X

PROLOGO

Cuando hace poco más de veinte años aparecía en el mercado el primer microprocesador, pocos podían pensar lo que los ordenadoreś iban a conseguir en tan breve período de tiempo. Hacía varios siglos ya que los chinos habían comenzado a trabajar con lo que hoy denominamos álgebra booleana o su equivalencia en sistema binario de numeración y sin embargo no fue hasta poco antes de la segunda guerra mundial cuando se rescató dicho sistema para el mundo tecnológico.

Fueron algunas universidades junto con la floreciente industria militar quienes comenzaron a buscar nuevas soluciones y mejoras sustanciales en los equipos y sistemas existentes en aquella fecha. Comenzó la era de la electrónica, aunque en niveles y proporciones no comparables con los actuales. Aparecía muy tímidamente y como un complemento de otros elementos una pequeña rama denominada electrónica digital. Esta rama pretendía que todos los elementos que formaran parte de un sistema funcionaran con la lógica más simple que se pudiera pensar, es decir, que funcionaran o no funcionaran, que circulara corriente o que no lo hiciera, etc., disponiendo siempre de dos opciones a las que se asignaban valores constantes de tensión.

En la década de los sesenta los circuitos integrados llegaron a los hogares en forma de calculadoras, relojes y productos sencillos de consumo, lo que proporcionó al campo digital una gran dosis de propaganda. Sin embargo la revolución llegó con el ordenador, no sólo por la aportación que individualmente realiza sino porque ha permitido su aplicación en casi todos los campos, tecnológicos o no, propiciando así la mejora de otros productos y abriendo líneas de trabajo, investigación y desarrollo tecnológico imposibles de realizar sin dicha herramienta.

Si hace unos años hablar de digital era mencionar un capítulo de un libro, ahora hablar de digital es hablar de toda una auténtica institución. La tecnología en general está basada en equipos digitales fundamentalmente, las enseñanzas universitarias en sus ramas técnicas, científicas y de ingenierías contienen asignaturas de contenido puramente digital, bien como circuitos digitales, sistemas digitales, comunicaciones, control, robótica, arquitectura de ordenadores, etc. Los equipos al alcance de investigadores y especialistas son cada vez mejores, más completos y sofisticados e incluso más baratos y asequibles dado el volumen tan elevado de unidades que se fabrican.

Las obras dedicadas a los circuitos digitales han proliferado muchísimo existiendo actualmente una gran bibliografía al respecto así como gran cantidad de información suministrada por los propios fabricantes de circuitos integrados y equipos. Tanta es la oferta y la demanda que en muchos campos los sistemas se quedan "anticuados" en un plazo no mayor de uno o dos años, de ahí que quienes están inmersos vocacional o profesionalmente en este campo y sus vertientes saben que deben renovar continuamente sus conocimientos.

Dentro del abanico de trabajos realizados sobre el campo digital, esta obra denominada FUNDAMENTOS DIGITALES trata de aportar una visión lo más realista posible de las principales enseñanzas y experiencias que tienen lugar en la docencia, aprendizaje y puesta en práctica de los conceptos básicos que definen la circuitería digital. Muchos libros ofrecen conceptos sacados de trabajos de investigación por lo que suelen quedar demasiado lejos del alcance de estudiantes y técnicos. Este trabajo conjuga información procedente de conceptos teóricos ya clásicos en la materia, desarrollos y estudios persona-

les del autor así como puntuales investigaciones prácticas sobre temas específicos, tratando de proporcionar en el conjunto de la obra una visión clara y concisa de los fundamentos básicos que como mínimo debe tener un futuro Ingeniero o Ingeniero Técnico y equivalentes de especialidades relacionadas con la electrónica, la informática y las comunicaciones.

Se analizan los sistemas básicos de numeración, su equivalencia con la lógica clásica booleana y sus formas de representación y optimización, proporcionando los medios necesarios para el diseño de los circuitos digitales más sencillos. Con el análisis de circuitos combinacionales integrados se consigue aumentar la complejidad y volumen de desarrollo, obteniendo unidades básicas de operación en los grandes sistemas digitales, como unidades de control, unidades aritmético-lógicas, memorias, etc. Como contrapartida a los dos primeros bloques genéricos descritos, el tercero está destinado al análisis de circuitos secuenciales elementales, precursores de la sincronización y del funcionamiento metódico, sistemático y automatizado. El bloque final es el circuito digital más importante en la actualidad, el microprocesador. Sus elementos constituyentes, funcionamiento y posibilidades dan una idea clara del potencial del mismo.

El aspecto más representativo de la obra es su continuo análisis mediante cronogramas, importantísima herramienta actual y futura en el diseño, descripción e información sobre cualquier circuito simple o complejo. Los analizadores lógicos son herramientas de captación y trascripción directa de resultados a cronogramas que cada vez se utilizan en mayor número, por lo que se ha pretendido dar a la obra no sólo el enfoque tradicional de una obra sobre la tecnología digital sino ofrecer una información que esté vigente en

los próximos años. Todos los temas por su contenido eminentemente práctico están profusamente analizados a través de numerosos ejemplos que deben dar una clara visión de la mayor parte de las posibilidades y opciones así como de gran cantidad de esquemas y figuras que junto con la teoría proporcionarán un adecuado marco de estudio.

Como quiera que actualmente estoy inmerso en un período no sólo de enseñanza sino de investigación mediante proyectos subvencionados por la Universidad de Cádiz y la Junta de Andalucía, espero que el proceso de generación de este trabajo sea una puerta abierta hacia nuevas y mejores obras en el campo de las nuevas tecnologías, robótica, automática, tecnología de computadores, sistemas digitales avanzados, etc.

El objetivo primordial de todo el esfuerzo realizado con esta publicación era proporcionar fundamentalmente a los alumnos de enseñanzas universitarias una obra de guía y consulta en el campo digital. Espero que a los alumnos de mi Universidad, especialmente a mis propios alumnos y a los de otras Universidades Andaluzas les pueda resultar útil y les sirva bien para salir de importantes dudas y lagunas conceptuales o para afianzar conocimientos.

Gracias a todos los que directa o indirectamente han tenido que ver con el trabajo realizado, con su publicación y su posterior distribución.

Javier Holgado

INDICE

TEMA 1

INTRODUCCION. ANALOGICO Y DIGITAL

1.1. Introducción
1.2. Diferencias entre analógico y digital
1.3. Ventajas e inconvenientes de los equipos digitales

1

ANALOGICO Y DIGITAL

1.1. INTRODUCCION

El presente tema trata a modo de comentario e información general, de introducir las primeras nociones sobre la palabra digital y sus orígenes. Se analizan los aspectos genéricos de su concepción, una breve historia de sus comienzos, y una idea genérica de la estructura de los elementos y soportes digitales como grandes dispositivos operativos. Aparecerá el concepto de computador, uno de los fines más importantes de la historia de las máquinas digitales, y se establecerá la comparación entre los equipos digitales y los analógicos, predecesores de los digitales, aunque la aparición de éstos últimos no ha supuesto la desaparición de los primeros, sino que durante bastantes años han seguido caminos paralelos. Se definirán breves ideas que darán un ligero conocimiento inicial de los computadores que se han usado y que se usan para realizar las operaciones de cálculo y de procesamiento de información.

1.2. DIFERENCIAS ENTRE ANALOGICO Y DIGITAL

Dependiendo de la gama de valores que pueden tomar las diversas magnitudes, se las puede clasificar en magnitudes analógicas y digitales. Las analógicas tienen un rango de variación continua, es decir, que en cualquier momento se puede obtener información sobre las mismas, existiendo valores en régimen de continuidad, sin puntos de desinformación. Las magnitudes digitales se denominan así porque sólamente pueden tomar valores discretos, es decir, la información que se puede obtener de ellas no es continua y se realiza mediante tomas puntuales, lo que implica que entre dos tomas consecutivas de información se desconoce realmente el valor de la magnitud

que se está estudiando. Este defecto de desinformación se suple mediante estimación de los valores que puede tomar en dichos intervalos y mediante reducción del tiempo de toma de información. La distinción entre ambas magnitudes es similar a la existente entre las longitudes de los segmentos, que son magnitudes analógicas continuas y los números naturales, que son magnitudes digitales discretas. Puede decirse que la primera es la línea continua y la segunda corresponde a algunos puntos de esa línea continua.

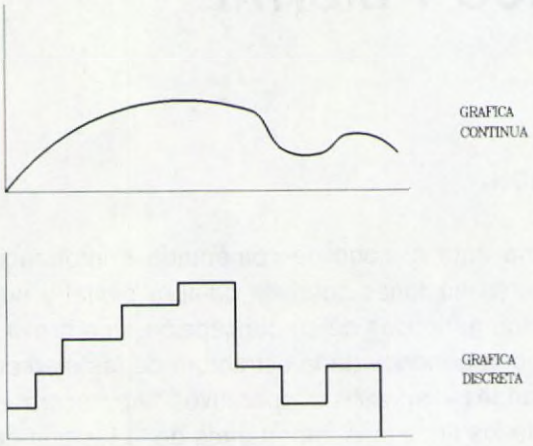


Figura 1.1. Gráficas continua y discreta

La denominación digital procede de dígito, denominación que recibe una cifra o número, ya que las magnitudes digitales como se verá posteriormente, pueden hacerse equivalentes a una representación numérica exacta mediante un número finito de cifras. Las magnitudes analógicas, al tener un rango continuo de variación solo podrán ser representadas numéricamente de ma-

Figura 1.2. Medición digital y analógica del tiempo

nera aproximada, aunque esta aproximación puede ser tan grande como se desee. Las denominaciones digital y analógico de la información son aplicables también a los equipos y máquinas que procesan dichas magnitudes, denominándose por ejemplo computadores digitales y computadores analógicos a los que procesan respectivamente cada una de dichas magnitudes. Las aplicaciones digitales son muy numerosas, aunque actualmente parece que el campo de los computadores es el que está tomando mayor auge.

Un computador analógico puede obtener la solución de una ecuación diferencial como la correspondiente al circuito eléctrico de la figura 1.3. Una vez conectado el interruptor de puesta en marcha PM circulará una corriente eléctrica determinada por:

$$U_b=(R+LS+1/CS)i \qquad [1.1]$$

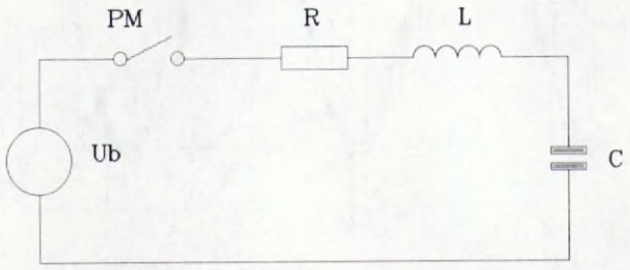


Figura 1.3. Circuito eléctrico

La precisión de la operación depende de la exactitud de cálculo del desarrollo en serie que determina el computador analógico. El computador digital no trabaja con cantidades sino con valores fijos (dígitos), pero su precisión es enormemente mayor y puede ser fácilmente programado para realizar cualquier tipo de operación y cálculo.

Cuando se investiga sobre el primer elemento operativo que sirvió para constituir los modernos equipos digitales, se coincide en que el antiguo ábaco que podía sumar y restar, podría ser considerado el primer sistema de cálculo operativo. El ábaco tenía como base la colocación de un grupo de "cuentas", de distintos valores cada una, insertadas en hilos colocados en un armazón de madera, que realizaban la operación de contar. Este proceso de

contar provenía de la utilización de pequeñas piedras o "calculus" en latín. De ahí el nombre asignado a esos pequeños objetos que ha perdurado a lo largo de los siglos. Pero el ábaco, no obstante, no ha tenido la consideración de dispositivo calculador, denominación que tradicionalmente se ha asignado en su primera concepción al instrumento inventado por Pascal a mediados del siglo XVII, constituido por un conjunto de ruedas dentadas, engranadas de tal forma que a cada vuelta de una de ellas hacía avanzar a la rueda de orden superior situada a su lado, acumulando resultados.

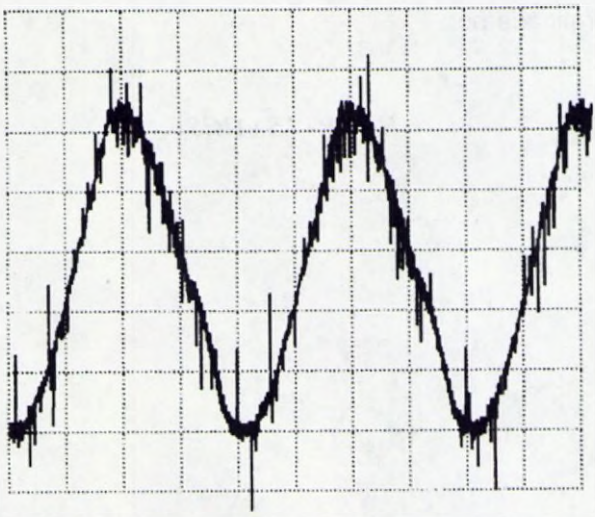


Figura 1.4. Señal analógica medida en osciloscopio

En la historia aparecen muchos nombres de máquinas y dispositivos, pero fue Hollerit quien en en el siglo XIX dio un gran impulso con un descubrimiento importantísimo dado que definió la base de los dispositivos externos de almacenamiento de información. Este equipo utilizaba información grabada mediante perforaciones en unas cartulinas denominadas tarjetas perforadas. La tarjeta perforada se ha empleado hasta el día de hoy, habiendo sido sustituida por los dispositivos de cinta perforada y posteriormente por los dispositivos de tipo magnético de almacenamiento de información, tales como discos, cintas, tarjetas, tambores, etc.

Como es lógico imaginar, las primeras máquinas de calcular disponían de una capacidad y unas posibilidades muy limitadas, pero con la aparición de elementos mecánicos, eléctricos y electrónicos a lo largo de los años, sus posibilidades fueron creciendo y de simples máquinas de calcular, se pasó a complejos, completos y muy sofisticados equipos de cálculo y procesado de

datos. Esta modernización de los equipos aumentando sus posibilidades, determinó que en la década de 1940 a 1950, aparecieran las primeras máquinas, a las que ya se les podía asignar la denominación de computadoras. Aunque fue Leibnitz en el siglo XVII quien detalló los aspectos relativos al antiguo sistema binario empleado por los chinos, Turing en 1936 comenzó a aplicarlo junto con la determinación del álgebra de Boole. Construyó una máquina de cálculo con cinta de almacenamiento, cabezal de lectura y escritura y unidad de control que ha servido como modelo para el desarrollo de numerosos equipos y programas de los actuales computadores.

El perfeccionamiento de los equipos con relés, permitió que se construyera la primera máquina totalmente automática por Howard H. Aiken, a quien puede asignarse la primera computadora digital de la historia. Esta máquina que fue denominada Marc-1, fue construida en la Universidad de Harvard en 1944. El tiempo de operación se redujo considerablemente una vez que se introdujeron los válvulas electrónicas en los computadores, ya en desuso en la actualidad al haber sido sustituidas por transistores y circuitos integrados. Como los computadores de este período no podían ser programados, sino que al realizar las instrucciones de ejecución de una operación ya no podían modificarse, fue necesario ampliar las posibilidades de los equipos diseñando la base de los futuros programas y de la consiguiente técnica de programación, con la aparición de los correspondientes lenguajes. Se diseñó entonces por parte de Von Neumann una máquina que podía definir unas instrucciones externas de funcionamiento del equipo, con una estructura similar a la de los actuales computadores. Dichas instrucciones se almacenaban en el interior de los equipos en unos dispositivos denominados memorias, apareciendo las primeras codificaciones de operaciones internas y el comienzo de lo que posteriormente se denominaría como lenguaje o código de máquina que constituyó la definición de la base operativa de funcionamiento de los computadores digitales mediante la asignación de una nomenclatura específica en sistemas de numeración binaria y hexadecimal.

En 1952 apareció el primer computador con memoria aunque los primeros computadores que se fabricaron en serie para ser comercializados fueron el UNIVAC-1 y el IBM-701, cuyas memorias centrales estaban constituidas por grandes tambores magnéticos. Posteriormente el modelo IBM 704 introdujo el núcleo de ferrita como elemento base de la constitución de sus memorias.

La aparición del circuito integrado en 1960 supuso para la electrónica digital una enorme potenciación de sus posibilidades de empleo, culminando en 1971 con la fabricación del primer microprocesador por parte de INTEL en un circuito integrado. Este elemento denominado unidad central de proceso ó UCP, es la parte más importante de un computador y constituye la base de diseño y construcción de los actuales sistemas digitales, siendo la parte más compleja e importante en los sistemas de procesamiento de información.

Las diferentes etapas de evolución tecnológica han ido determinando las generaciones de computadores. Haciéndolos casi coincidir con las décadas, se denominó primera generación a los primeros computadores construidos con válvulas de vacío hasta los años 50. A partir de 1950 con la aparición del transistor, se constituye la segunda generación, reduciéndose el tamaño, el coste y el consumo.

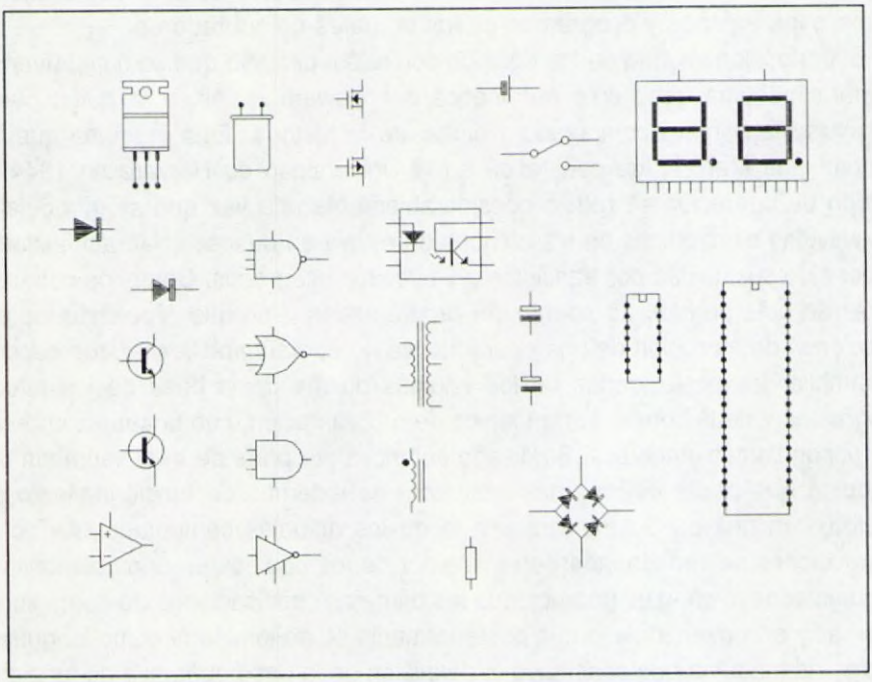


Figura 1.5. Elementos de los equipos digitales

En la década de los 60, con la aparición del circuito integrado se comienza la tercera generación, surgiendo nuevos conceptos como programación, teleproceso y la aparición de las memorias electrónicas de semiconductores, con mayor capacidad de almacenamiento de datos. La aparición del microprocesador en los 70 determina la cuarta generación, aumentando el número de circuitos en el interior de los integrados, reduciendo el número de componentes y facilitando su aplicación en robótica, control de procesos industriales, instrumentación, etc.

En los años 80, el aumento de la escala de integración y la introducción del concepto de inteligencia artificial definen los computadores de la quinta generación. Los años 90 se enfocan desde las aplicaciones de los sistemas expertos, el aumento de prestaciones de los sistemas digitales, el aumento

de la capacidad de memoria en los circuitos integrados y la potenciación de las nuevas arquitecturas.

Con la creación de estos equipos se determinó no sólo el comienzo de una nueva línea de desarrollo técnico, sino que al mismo tiempo fijaron unos estándares de producción que se mantienen hoy día a pesar de los años transcurridos, tratándose cada vez más de homogeneizarlos y hacerlos compatibles para conseguir aumentar las posibilidades de los mismos.

1.3. VENTAJAS E INCONVENIENTES DE LOS EQUIPOS DIGITALES

La velocidad de respuesta muy alta en los equipos digitales, especialmente en los que pueden simultanear varias tareas se destaca como aspecto clave frente a la velocidad media de los analógicos, que llega a ser bastante lenta en los equipos que tienen conectados un gran número de elementos de cálculo. Al mismo tiempo los actuales equipos digitales ofrecen la posibilidad de operar en un intervalo muy amplio de valores, pueden operar en coma flotante y disponen de registros de almacenamiento de gran capacidad.

El equipo digital opera mediante secuencia de instrucciones que realizan determinadas operaciones, almacenando resultados intermedios en las memorias de los sistemas para obtener un resultado final con mayor velocidad. El analógico, por sus elementos operativos debe realizar simultáneamente la obtención de un determinado cálculo. En los equipos digitales, a la vista de las mejoras que se introducen día a día, se podría decir que las posibilidades son casi ilimitadas, o que al menos no se ha determinado todavía un techo para los mismos. Como en los analógicos las operaciones dependen de los elementos que los constituyen, sus posibilidades sí están muy limitadas, habiéndose centrado con el paso de los años en la resolución de ecuaciones diferenciales.

El equipo digital responde con gran velocidad lo que determina un tiempo de respuesta muy breve, aumentando estas posibilidades con la aparición de nuevos equipos con frecuencias internas de funcionamiento cada vez más elevadas, lo que redunda en tiempos de cálculo cada vez inferiores. El tiempo de respuesta de un equipo analógico es relativamente elevado, especialmente cuando se compara con el tiempo de un equipo digital. Depende de los elementos de cálculo que tenga conectados y del número de operaciones que tenga que realizar simultáneamente.

Las posibilidades de ampliación del equipo digital son enormes en cualquier campo de aplicación que se desee. La fabricación y el diseño de nuevos elementos de "hardware" y "software" hacen que las aplicaciones de estos equipos crezcan cada día más. En cambio, las aplicaciones de los analógicos está limitada a los elementos existentes en la actualidad, no te-

niendo excesivas posibilidades dado que el campo digital abarca en estos momentos todas las expectativas de diseño y de nuevas creaciones.

La velocidad de adaptación a nuevos cambios es elevada en los digitales, dado que al trabajar con programas o "software", se pueden preparar cambios en los mismos mientras se está trabajando, realizando su posterior implantación de manera casi inmediata. Los analógicos, dado que normalmente se hace necesario la desconexión de los mismos, tienen una velocidad lenta de reacción ante cambios en su estructura.

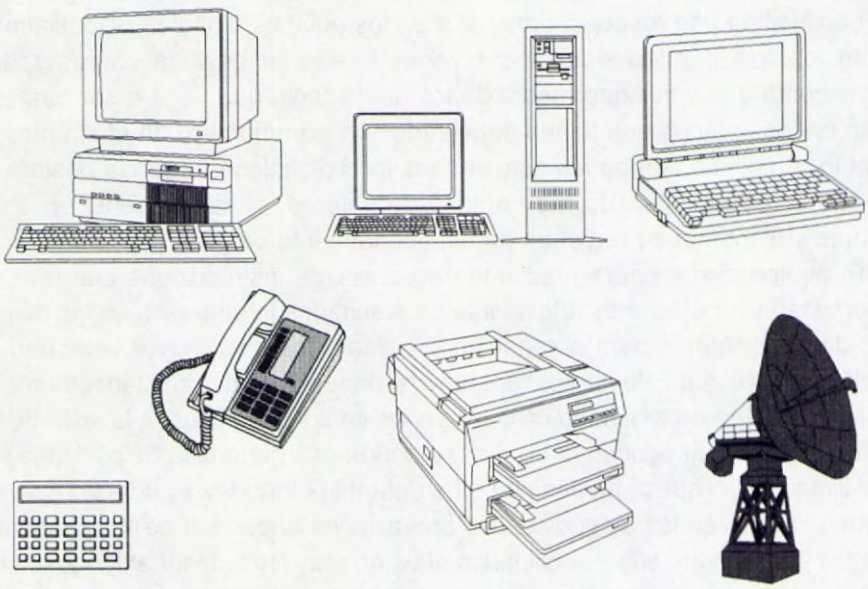


Figura 1.6. Aplicaciones digitales

La inversión en equipos digitales era elevada en sus comienzos, especialmente si se conectaban elementos periféricos que facilitaran información o cálculos paralelos, aunque la gran demanda de los mismos ha hecho que la fabricación en serie de grandes cantidades de equipos haya abaratado considerablemente los precios. Las actuales industrias de semiconductores y cadenas de montaje en Japón, Corea, Taiwan y otras zonas orientales ha conseguido que los equipos digitales en general y especialmente la electrónica digital de consumo alcancen unos precios cada vez más bajos con un gran nivel de calidad.

Teniendo en cuenta las características descritas anteriormente, los equipos digitales están en pleno auge, disponiendo de inmensas posibilidades en el futuro para todo tipo de aplicaciones, algunas de las cuáles aún ni se han empezado a investigar. En los analógicos en cambio, las posibilidades son muy reducidas, dado que actualmente cualquier operación realizada por dichos equipos puede ya ser simulada con gran precisión en los equipos digitales, viéndose reducidas sus posibilidades futuras a campos muy particulares de la técnica y la investigación. Las aplicaciones digitales en relojería, calculadoras, instrumentos de medida, automatización, investigación espacial, medicina, proyectos militares, y especialmente computadores, periféricos, comunicaciones, etc., han conseguido generar una de las más importantes herramientas de trabajo.

TEMA 2
EL SISTEMA BINARIO

2

EL SISTEMA BINARIO

2.1. INTRODUCCION

El sistema binario constituye la base de funcionamiento de un sistema digital, basándose en la dualidad de valores "0" y "1" que puede tomar. Se pueden representar de esta forma los valores de tensión empleados en los circuitos integrados. Haciendo equiparar los valores de existencia o no de tensión con los valores binarios, se pueden analizar de forma muy sencilla los circuitos electrónicos que se definen como circuitos digitales.

El desarrollo del tema se centra en la presentación del sistema binario de numeración y sus relaciones con los sistemas decimal, octal y hexadecimal, conversiones, agrupaciones y aplicaciones. Estos sistemas, especialmente el hexadecimal, es enormemente utilizado como traductor de la información que se procesa en los equipos informáticos. Asimismo se verá la agrupación de bits del sistema binario, definiendo las cifras codificadas (BCD). Como herramienta importantísima de aplicación práctica en circuitos digitales se presentan los cronogramas, que basados en el estudio de señales de tiempo periódicas y aperiódicas informan visualmente (gráficamente, mediante osciloscopio o con analizador lógico) de las señales que se procesan y generan en los circuitos. Estos cronogramas serán de aplicación a todos los elementos digitales que se analizan en los temas posteriores.

2.2. EL SISTEMA DE BASE 2. CONVERSION A DECIMAL

Un sistema de numeración está formado por un conjunto simple y pequeño de cifras básicas, que agrupadas y combinadas entre sí dan lugar a la formación de cualquier expresión numérica. En concreto, el sistema decimal

empleado usualmente se define así por estar formado por diez cifras, del 0 al 9, que constituyen los elementos simples que formarán parte de todos los números que se expresen en este sistema. El sistema binario, siguiendo la misma concepción que el decimal y que los demás sistemas de numeración, está constituido por 2 cifras básicas (sistema de base 2), el "0" y el "1". Cualquier expresión numérica que se realice en el sistema binario, estará formada exclusivamente por ceros y unos, como por ejemplo el "011011".

Cada cifra o **DIGITO** del sistema binario se suele denominar **BIT**. Esta expresión comúnmente utilizada en el lenguaje informático, sirve como referencia para la determinación de los equipos y sistemas de proceso de datos, dado que el número de elementos simples o bits que un sistema puede procesar simultáneamente, son la respuesta más clara de la capacidad y las posibilidades del mismo. Un número entero de cualquier sistema de numeración se representa mediante una ecuación en forma de desarrollo de potencias, tal como la siguiente:

$$M=a_{n-1}B^{n-1}+a_{n-2}B^{n-2}+\ldots+a_1B^1+a_0B^0=\sum_{i=0}^{n-1}a_iB^i \qquad [2.1]$$

M=número entero que se representa
B=base del sistema de numeración
$a_0,\ldots,a_{n-1}$=Coeficientes del número "M"

La base B equivale al número de cifras constitutivas del sistema que se emplea; por lo que se corresponde con 10 en el sistema decimal y 2 en el sistema binario. Sustituyendo en la ecuación [2.1] el valor de la base a utilizar, se obtiene el desarrollo correspondiente al número y sistema que se utilicen. La suma de términos de los desarrollos en potencias conduce a la obtención de un valor numérico decimal independientemente de que existan números con parte entera y fraccionaria. Este proceso se denomina **CONVERSION BINARIO-DECIMAL**. El desarrollo realizado por la ecuación [2.1] para números enteros, se aplica también para números fraccionarios, conduciendo a la siguiente expresión:

$$P=b_1B^{-1}+b_2B^{-2}+\ldots+b_{m-1}B^{-m+1}+b_mB^{-m}=\sum_{j=1}^{m}b_jB^{-j} \qquad [2.2]$$

P=número fraccionario que se representa
B=base del sistema de numeración
$b_1,\ldots,b_m$=Coeficientes del número "P"

La agrupación de las ecuaciones [2.1] y [2.2] conduce a la obtención de una única ecuación genérica para el desarrollo de cualquier número con parte entera y fraccionaria en cualquier sistema de numeración:

$$N=a_{n-1}B^{n-1}+...+a_0B^0+b_1B^{-1}+...+b_mB^{-m}=\sum_{i=0}^{n-1}a_iB^i+\sum_{j=1}^{m}b_jB^{-j} \qquad [2.3]$$

N=número que se representa

B=base del sistema de numeración

$a_0,...,a_{n-1}$=Coeficientes enteros del número "N"

$b_1,...,b_m$=Coeficientes fraccionarios del número "N"

Generalizando, la conversión a decimal de un número en una base B cualquiera se realiza mediante la suma de los términos de su desarrollo en potencias.

*## **Ejemplo 2.1**: Representar en forma de ecuación de potencias el número decimal 3987.*

Solución:

$$3987_{10}=3*10^3+9*10^2+8*10^1+7*10^0$$

*## **Ejemplo 2.2**: Representar en forma de ecuación de potencias el número binario "101101".*

Solución:

$$101101_2=1*2^5+0*2^4+1*2^3+1*2^2+0*2^1+1*2^0$$

$$101101_2=2^5+2^3+2^2+2^0=45_{10}$$

*## **Ejemplo 2.3**: Representar en forma de ecuación de potencias el número decimal 0,187.*

Solución:

$$0,187_{10}=1*10^{-1}+8*10^{-2}+7*10^{-3}$$

*## **Ejemplo 2.4**: Representar en forma de ecuación de potencias el número binario 0,1101.*

Solución:

$$0,1101_2=1*2^{-1}+1*2^{-2}+1*2^{-4}=0,185_{10}$$

*## **Ejemplo 2.5**: Convertir a decimal el número binario 1101,011.*

Solución:

$$1101,011_2=1*2^3+1*2^2+1*2^0+1*2^{-2}+1*2^{-3}=13,375_{10}$$

2.3. CONVERSION A BINARIO DE UN NUMERO DECIMAL

El proceso opuesto al realizado en el apartado anterior es la conversión a binario de un número decimal. Se ejecuta de forma diferente el proceso para números enteros y fraccionarios. Las operaciones necesarias para realizarlo se detallan a continuación.

a) Proceso de conversión de **NUMEROS DECIMALES ENTEROS**:

* Se divide el número decimal por 2
* El **RESTO** corresponde al coeficiente de la potencia cero del número binario
* Se divide el cociente obtenido por 2
* El **RESTO** corresponde al coeficiente de la potencia uno del número binario
* Se siguen dividiendo consecutivamente los cocientes que se vayan obteniendo
* Los **RESTOS** que se obtengan serán los sucesivos coeficientes del número binario
* El último **COCIENTE** será el bit más significativo

El proceso de división finaliza al obtener un cociente de valor inferior a 2, que ya no podrá ser dividido más, siendo por tanto ese cociente el coeficiente de la última potencia del número binario, la más significativa (La cifra menos significativa es la situada a la derecha y la más significativa es la situada a la izquierda).

b) Proceso de conversión de **NUMEROS DECIMALES CON PARTES ENTERA Y FRACCIONARIA**:

* La parte entera se procesa tal como se indica en el apartado anterior
* La parte fraccionaria se procesa mediante sucesivos productos por 2
* Los bits de **PARTE ENTERA** que se obtienen de dichos productos son los sucesivos coeficientes de las potencias de base 2
* Los dígitos de la parte entera no se tienen en cuenta al efectuar la multiplicación siguiente
* Dependiendo del número de elementos o coeficientes de la parte fraccionaria que se deseen, se continuará realizando el proceso

Depende de cada operación y de cada operador el hecho de considerar suficientemente aproximado un número determinado de dígitos fraccionarios. La estimación se debe realizar igual que se realizaría en caso de operar con el sistema decimal (Se puede considerar una buena aproximación la realizada con seis dígitos de parte fraccionaria). Como ejemplo de números de bastante uso, se relacionan en la tabla 2.1 los primeros números que se pueden obtener del sistema binario, con su correspondiente valor en el sistema decimal.

BINARIO	DECIMAL
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

Tabla 2.1. Sistema binario

Ejemplo 2.6: *Convertir a binario el número decimal 157.*

Solución:

DIVISION	COCIENTE	RESTO
157/2	78	1
78/2	39	0
39/2	19	1
19/2	9	1
9/2	4	1
4/2	2	0
2/2	1	0

Resultado final: $\mathbf{157_{10}=10011101_2}$

Ejemplo 2.7: *Convertir a binario el número decimal 19,625.*

Solución:

Parte entera:

DIVISION	COCIENTE	RESTO
19/2	9	1
9/2	4	1
4/2	2	0
2/2	1	0

Parte fraccionaria:	**PRODUCTOS**	**RESULTADOS**
	0,6875*2	1,3750
	0,3750*2	0,7500
	0,7500*2	1,5000
	0,5000*2	1,0000

Resultado final: $19{,}6875_{10}=10011{,}1011_2$

2.4. SISTEMA DE REPRESENTACION OCTAL

El sistema octal es junto con el hexadecimal y el binario, una de las bases de numeración más utilizadas en sistemas digitales. Suele aplicarse como herramienta de representación de datos binarios de gran extensión al reducir el tamaño de la información a visualizar (octeto), especialmente en manejo de archivos en discos magnéticos. Tiene como elementos básicos las cifras decimales del 0 al 7, con su mismo valor y significado, pudiendo expresarse cualquier número de dicho sistema mediante la ecuación genérica [2.3] particularizada para la base B=8. El desarrollo y suma de los elementos de esta ecuación conduce a la **CONVERSION OCTAL-DECIMAL.**

La **CONVERSION DECIMAL-OCTAL** se realiza de igual forma que con el sistema binario, particularizando en este caso mediante divisiones o multiplicaciones sucesivas por 8 (la base B=8) según que se utilicen términos enteros o fraccionarios. En la tabla 2.2 se representan como ejemplo los primeros números del sistema octal.

OCTAL	DECIMAL
00	0
01	1
02	2
03	3
04	4
05	5
06	6
07	7
10	8
11	9
12	10
13	11
14	12
15	13
16	14
17	15

Tabla 2.2. Sistema octal

2.4.1. **Relación entre sistemas binario y octal**

En numerosas ocasiones, resultará más cómodo pasar un número directamente de binario a octal y viceversa sin tener que pasar por la etapa intermedia del sistema decimal, tal como lo realizan los equipos digitales, que almacenan la información siempre en binario y la traducen a octal para su más fácil comprensión y manipulación, por lo que se hace necesario buscar una equivalencia entre ambos sistemas, cosa que no será difícil dado que son sistemas múltiplos. Si el sistema octal se basa en las combinaciones de sus ocho cifras elementales, éstas las podemos asimilar con el número de combinaciones que pueden formarse con 3 bits del sistema binario. En la tabla 2.3 se representan las expresiones binarias de las ocho cifras elementales del sistema octal.

De esta forma, y usando los valores de la tabla 2.3, cualquier número en sistema octal se puede expresar en binario, sustituyendo cada cifra octal por su equivalente en dígitos binarios. Igualmente, si se quiere expresar un número binario en octal, se dividen los dígitos binarios en grupos de 3 bits, a partir de la coma hacia la derecha e izquierda, **COMPLETANDO CON CEROS LOS HUECOS** que pudieran existir.

OCTAL	BINARIO
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Tabla 2.3. Equivalencia entre sistemas binario y octal

*## **Ejemplo 2.8**: Convertir a octal el número decimal 610,1875.*

Solución:

Parte entera:

DIVISION	COCIENTE	RESTO
610/8	76	2
76/8	9	4
9/8	1	1

Parte fraccionaria:

PRODUCTOS	RESULTADOS
0,1875*8	1,5000
0,5000*2	4,0000

Resultado final:

$$610{,}1875_{10}=1142{,}14_{8}$$

*## **Ejemplo 2.9**: Convertir a decimal el número octal 724,15.*

Solución:

$$724{,}15_{8}=7*8^{2}+2*8^{1}+4*8^{0}+1*8^{-1}+5*8^{-2}=468{,}203125_{10}$$

*## **Ejemplo 2.10**: Convertir a binario el número octal 614,135.*

Solución:

$$614{,}135_{8}=110001100{,}001011101_{2}$$

*## **Ejemplo 2.11**: Convertir a octal el número binario 101101011101,1011001.*

Solución:

$$101101011101{,}1011001_{2}=5535{,}544_{8}$$

*## **Ejemplo 2.12**: Convertir a binario el número octal 1250,4.*

Solución:

$$1250{,}4_{8}=001010101000{,}100_{2}$$

*## **Ejemplo 2.13**: Convertir a octal el número binario 11111110110001,01001.*

Solución:

$$11111110110001{,}01001_{2}=37661{,}22_{8}$$

2.5. - SISTEMA DE REPRESENTACION HEXADECIMAL

Este sistema tiene como base de numeración dieciséis elementos simples, constituídos por las cifras decimales del 0 al 9 y las letras A, B, C, D, E, F. El planteamiento general de conversiones y desarrollos introducidos para el sistema binario, es también de aplicación a este sistema. Por ello, la ecuación general [2.3] determina su desarrollo en términos de potencia así como la **CONVERSION HEXADECIMAL-DECIMAL** mientras que la división o multiplicación por la base B=16 posibilita la **CONVERSION DECIMAL-HEXADECIMAL**. Este sistema, al representar algunos números mediante su equivalente en letras, necesita una conversión particular en las operaciones aritméticas entre letras y números. En la tabla 2.4 se representa la equivalencia de los elementos básicos del sistema hexadecimal.

HEXADECIMAL	DECIMAL
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15

Tabla 2.4. Equivalencia del sistema hexadecimal

Este sistema de representación numérica es enormemente utilizado en equipos informáticos debido a la reducción tan considerable de espacio y a la comodidad de su manipulación frente al sistema binario. Se puede comprobar en cualquier ordenador personal, disponiendo del sistema operativo ade-

cuado o de un software de utilidades, cómo la información binaria contenida en cualquier soporte (por ejemplo una unidad de diskette) es expresada mediante sistema hexadecimal. Decodificando adecuadamente dicha información hexadecimal, se puede obtener la información real grabada.

2.5.1. **Relación entre sistemas binario y hexadecimal**

Del mismo modo que se ha analizado anteriormente la relación entre los sistemas binario y octal, se puede realizar entre los sistemas binario y hexadecimal, dada la gran utilización de ambos sistemas. Es de resaltar la relación de ambos sistemas a través de las estructuras básicas de los actuales microprocesadores, de 8, 16, 32 bits, fácilmente agrupables en 2, 4 y 8 elementos hexadecimales. La relación se realizará ahora mediante las 16 posibles combinaciones de 4 bits del sistema binario, representadas en la tabla 2.5. Cualquier número hexadecimal se representa en binario sustituyendo cada cifra por su equivalente binario con 4 bits, mientras que la expresión hexadecimal de un número binario se realiza mediante división en grupos de 4 bits, a partir de la coma tanto hacia la derecha como hacia la izquierda, **COMPLETANDO CON CEROS LOS HUECOS** que pudieran existir.

HEXADECIMAL	BINARIO
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Tabla 2.5. Equivalencia entre sistemas binario y hexadecimal

*## **Ejemplo 2.14**: Convertir a hexadecimal el número decimal 953,625.*

Solución:

Parte entera:

DIVISION	COCIENTE	RESTO
953/16	59	9
59/16	3	11

Parte fraccionaria:

PRODUCTOS	RESULTADOS
0,625*16	10,000

Resultado final:

$$953{,}625_{10}=3B9{,}A_{16}$$

*## **Ejemplo 2.15**: Convertir a decimal el número hexadecimal F31,C8.*

Solución:

$$F31{,}C8_{16}=15*16^2+3*16^1+1*16^0+12*16^{-1}+8*16^{-2}=3889{,}78125_{10}$$

*## **Ejemplo 2.16**: Convertir a binario el número hexadecimal B72,F4.*

Solución:

$$B72{,}F4_{16}=101101110010{,}11110100_2$$

*## **Ejemplo 2.17**: Convertir a hexadecimal el número binario 11101011001110,01001.*

Solución:

$$11101011001110{,}01001_2=3ACE{,}48_{16}$$

*## **Ejemplo 2.18**: Convertir a binario el número hexadecimal CABE,14.*

Solución:

$$CABE{,}14_{16}=1100101010111110{,}00010100_2$$

*## **Ejemplo 2.19**: Convertir a hexadecimal el número binario 10100001111111101010,1.*

Solución:

$$10100001111111101010{,}1_2=A1FEA{,}8_{16}$$

2.6. **EL CODIGO BCD**

Expresar un número decimal en el código binario, puede suponer en el caso de números de elevado valor decimal o con gran cantidad de cifras en su parte fraccionaria que el número binario natural obtenido tenga una gran cantidad de bits. Esto hace que la representación en sistema binario natural esté bastante limitada, y se tienda a buscar otros sistemas de representación con menor limitación y mayor facilidad de cálculo operativo. Con las siglas BCD, se abrevia la expresión "Decimal codificado en binario", en su transcripción inglesa. Consiste en la agrupación de 4 bits, con diferentes variantes según el sistema que se emplee.

Básicamente se definen tres posibilidades de codificación en el código BCD que se denominan BCD natural, BCD Aiken y BCD Exceso-3. Existen otras agrupaciones de bits, con base en el sistema BCD, pero que no tiene prácticamente relevancia, por lo que se omite su análisis. Cada sistema tiene sus ventajas e inconvenientes y sus aplicaciones particulares.

2.6.1. **El código BCD NATURAL**

Se denomina de esta forma a una serie de combinaciones obtenidas con las cuatro primeras potencias enteras del sistema binario, es decir, las potencias 0, 1, 2 y 3. Con estas cuatro potencias, se pueden obtener dieciséis combinaciones de sus coeficientes, de los cuáles sólo se utilizan diez. Corresponden a las diez primeras combinaciones de dichos bits, que coinciden con los diez números del sistema decimal, del 0 al 9. Puede por tanto decirse que el sistema codificado en BCD natural se corresponde con la conversión al sistema binario con 4 bits, de las diez primeras cifras de dicho sistema, según se observa en la tabla 2.6. Este sistema ofrece una conversión directa con el sistema decimal pero en cambio posee combinaciones binarias no utilizadas, lo que producirá inconvenientes en la aritmética.

2.6.2. **El código BCD AIKEN**

Se denomina así a una versión particular del código BCD natural, en la cuál la potencia de orden 3 se sustituye por una potencia de orden 1, con lo cuál se tendrán cuatro potencias, de orden 0, 1, 2 y 1. De esta forma, el valor máximo de las combinaciones que se pueden formar con los coeficientes de dichas potencias, coincide con el 9, valor máximo de los elementos simples del sistema decimal. El sistema BCD Aiken es autocomplementario, porque al

sustituir los ceros por unos y los unos por ceros en cada una de sus combinaciones, se obtiene el complemento a nueve de dichas cifras, estando los mismos incluidos en la propia tabla como se puede observar en 2.7. (Complemento a 9 es la diferencia entre 9 y el número que se utiliza). Ofrece como ventaja la utilización de todas las combinaciones pero rompe la estructura del sistema de numeración y no siendo por tanto compatible directamente con otros sistemas.

BCD NATURAL	DECIMAL
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9

Tabla 2.6. Código BCD natural

BCD AIKEN	DECIMAL
0000	0
0001	1
0010	2
0011	3
0100	4
1011	5
1100	6
1101	7
1110	8
1111	9

Tabla 2.7. Código BCD Aiken

2.6.3. **El código BCD EXCESO-3**

Consiste en la adición de 3 unidades binarias a todos los elementos del código BCD natural, de ahí la expresión y denominación de este código de numeración. Su configuración básica es la misma, pero habrá que operar de forma diferente cuando se utilice en operaciones aritméticas, como se verá más adelante. Es también un código autocomplementario, ya que cualquier complemento que se realice pertenece también al sistema. En la tabla 2.8 se indican sus elementos y su relación con el sistema decimal.

BCD EXCESO-3	DECIMAL
0011	0
0100	1
0101	2
0110	3
0111	4
1000	5
1001	6
1010	7
1011	8
1100	9

Tabla 2.8. Código BCD Exceso-3

2.7. **CONVERSION DECIMAL-BCD**

Un número decimal puede representarse en cualquiera de los códigos BCD mediante la sustitución de cada una de las cifras decimales por su equivalente en BCD. De esta manera, mientras el aumento del valor absoluto del número decimal suponía en el sistema binario el aumento del número de bits, en el sistema BCD se sigue manteniendo el número de bits si se mantiene el número de cifras decimales.

*## **Ejemplo 2.20**: Convertir a los tres códigos BCD el número decimal 328,51.*

Solución:

Expresión en BCD natural:

$$328{,}51_{10}=001100101000{,}01010001_{BCD\text{-}N}$$

Expresión en BCD Aiken:

$$328{,}51_{10}=001100101110{,}10110001_{BCD\text{-}A}$$

Expresión en BCD-Exceso 3:

$$328{,}51_{10}=011001011011{,}10000100_{BCD\text{-}E3}$$

2.8. EL CODIGO PROGRESIVO DE GRAY

Se denominan códigos progresivos aquellos que entre combinaciones binarias consecutivas sufren variación únicamente en un bit, es decir, se mantienen constantes todos los dígitos excepto uno. El más representativo de los códigos progresivos es el código reflejado o de Gray, denominado así por poseer simetría la tabla de combinaciones al aumentar el número de dígitos. A medida que aumenta el número de bits se duplica la tabla en forma simétrica, añadiéndole a la primera mitad el "0" como nuevo bit, y el "1" a la segunda mitad. Así se forma la tabla 2.9 construida con 4 bits.

Este tipo de código se utiliza sobre todo en sistemas de control donde no se permiten cambios de más de un bit que puedan inducir al sistema a cometer errores. Así por ejemplo, según el esquema de la figura 2.1 correspondiente a una cadena de producción donde existen contactos eléctricos que determinan el código de la operación a realizar, no se pueden realizar saltos entre operaciones de más de un bit ya que si estando en el código 101 se deseara pasar al 010, podría producirse la evolución a través de 101-100-110-010 (una posibilidad), lo que provocaría un fallo de fabricación pues se realizarían operaciones no previstas en las secuencias productivas.

2.9. BIT DE SIGNO

Los números de cualquier sistema de numeración deberán llevar la misma estructura que el sistema decimal, por lo que además de la conversión o

representación correspondiente al módulo del número que se analiza, será necesario indicar el signo del mismo. Como sólo se necesitan 2 valores que representen los signos positivo y negativo, se utiliza el **CONVENIO POSITIVO** (lógica positiva) de representación, que asigna a los números positivos el valor "0" y a los negativos el valor "1", situando dicho bit delante de la cifra a representar, en la posición más significativa. (La lógica negativa asigna los valores opuestos)

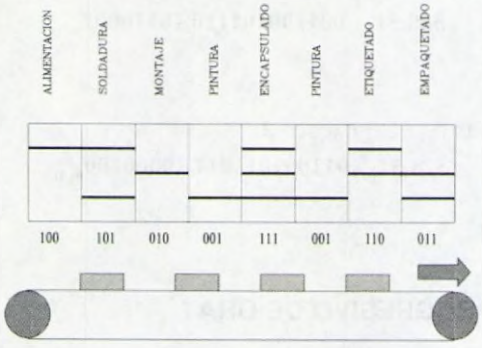


Figura 2.1. Caso práctico de código progresivo

CODIGO GRAY	DECIMAL
0000	0
0001	1
0011	2
0010	3
0110	4
0111	5
0101	6
0100	7
1100	8
1101	9
1111	10
1110	11
1010	12
1011	13
1001	14
1000	15

Tabla 2.9. Código de Gray

Así por ejemplo, el número +15 en decimal se representa como 01111 en binario, mientras que el -15 se corresponde con el 11111. Este bit de signo es de aplicación a todos los sistemas de numeración empleados en equipos digitales (binario, octal, hexadecimal, BCD, etc.), excepto en aquellos casos en que el sistema digital reserve forzosamente más espacio para almacenar el bit de signo o lo codifique de forma particular.

En los registros de almacenamiento de los equipos informáticos, el bit de signo ocupa siempre la posición más significativa, quedando el resto de los bits para almacenar el valor absoluto del dato. Así por ejemplo, el número -15 se representa en un registro de 8 bits como el 10001111. (Esta expresión genérica del bit de signo a veces aparece desvirtuada cuando algunos equipos digitales almacenan directamente los números negativos mediante complementos o en formato de coma flotante).

2.10. REPRESENTACION GRAFICA DE LAS SEÑALES DIGITALES. CRONOGRAMAS

Una señal digital se corresponde con una señal trapezoidal como la de la figura 2.2, teniendo en cuenta que no existen perturbaciones ni ruidos. En ella se representan los niveles de tensión eléctrica mediante valores binarios "0" y "1" y de la que se desprenden importantes aspectos para los sistemas digitales, tales como flancos, niveles, bandas, frecuencia, período, etc. Esta señal genérica está analizada en lógica positiva, asignando un "1" a la existencia de señal y "0" a la ausencia de la misma.

En las situaciones físicas reales, los tiempos de activación y desactivación no son nulos (respuesta instantánea), sino que aunque mínimos, existen

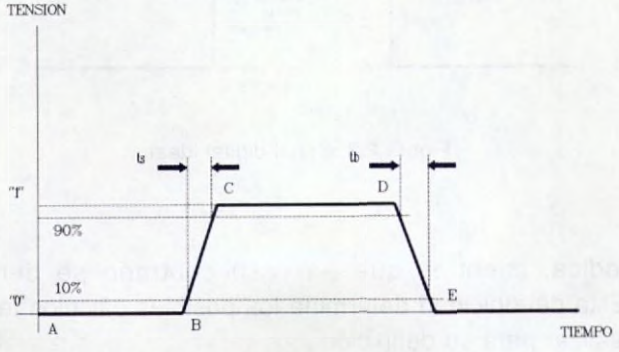


Figura 2.2. Señal digital

unos intervalos que emplearán los diversos componentes electrónicos que constituyen los circuitos integrados en alcanzar los valores que se hayan determinado. De ahí que el cambio entre niveles no sea inmediato.

Denominando **FLANCOS** a los tramos B-C y D-E y **NIVELES** a los tramos A-B y C-D, se define como nivel cero lógico ó "0" al tramo A-B y como nivel uno lógico ó "1" al tramo C-D. Se pasa de un nivel a otro mediante flancos (de subida y de bajada), pero sólo se considera estable el período comprendido entre el 10% y el 90% de la señal, representado por t_s y t_b, y por tanto útil.

Existen circuitos que se activan cuando aparecen los flancos (activación por flancos) y otros que lo hacen cuando se estabilizan las señales en los niveles correspondientes (activación por nivel). Una secuencia de números binarios equivale a una secuencia de señales digitales trapezoidales como las de la figura 2.2. Sin embargo, debido al error tan pequeño que se produce al considerar que los flancos son perpendiculares a las líneas de nivel, se suelen representar las señales digitales mediante una aproximación ideal como la indicada en la figura 2.3.

La representación de las señales digitales se realiza mediante diagramas tensión/tiempo o **CRONOGRAMAS**. Si la señal se repite a lo largo del tiempo, es periódica, mientras que en caso contrario se denomina señal aperiódica. Esta periodicidad determina los posibles cálculos temporales que se pueden realizar para su definición.

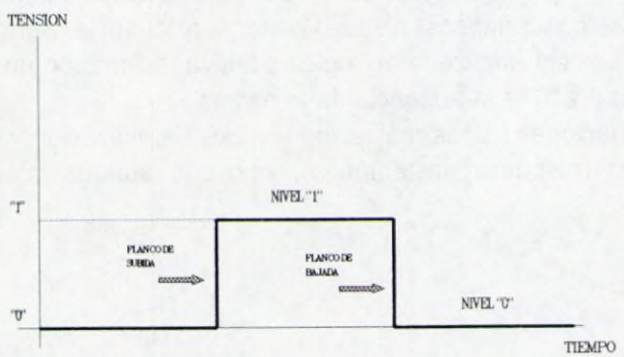


Figura 2.3. Señal digital ideal

Así por tanto, la duración de una señal digital o **PERIODO**, expresada en segundos, se obtiene de la siguiente ecuación:

$$T=1/F \qquad [2.4]$$

La **FRECUENCIA** es inversa del período, su unidad básica de medida es el Hertzio (Hz) y tiene por expresión:

$$F=1/T \qquad [2.5]$$

Si la señal es aperiódica, su representación equivale a la de la figura 2.4 (a) y si por el contrario es una señal periódica, se repetirá cíclicamente tal como indica el cronograma de la figura 2.4 (b). Una señal de este tipo suele denominarse **SEÑAL RELOJ** o **CLOCK**, y es muy importante en el estudio de la sincronía de los componentes digitales.

Existen equipos de medida que permiten comprobar las señales digitales que se están generando en los equipos. Entre los más útiles y conocidos están las sondas lógicas (observación no gráfica), los osciloscopios (observación visual de señales periódicas) y sobre todo los analizadores lógicos (observación visual de todo tipo de señales digitales). En las figuras 2.5 y 2.6 se muestran imágenes típicas obtenidas en las pantallas de un osciloscopio y un analizador lógico respectivamente.

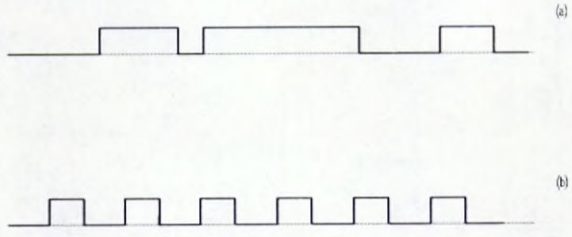


Figura 2.4. Cronogramas

*## **Ejemplo 2.21**: Calcular la frecuencia de una señal digital con período 1^{ms} (milisegundo).*

Solución:

$$F=1/T=1/0{,}001=1000^{Hz}$$

*## **Ejemplo 2.22**: Calcular el período de una señal digital de frecuencia 1^{MHz} (megahertzio).*

Solución:

$$T=1/F=1/1000000=0{,}000001^{s}=1^{\mu s}$$

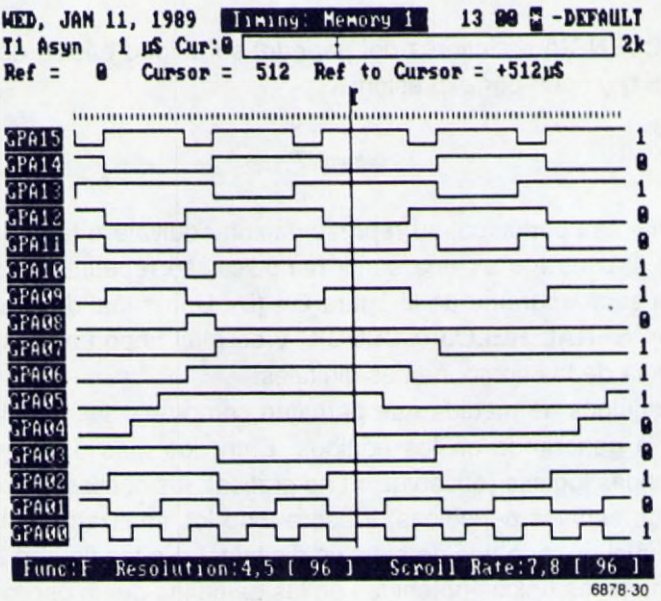


Figura 2.5. Señal de reloj medida en osciloscopio

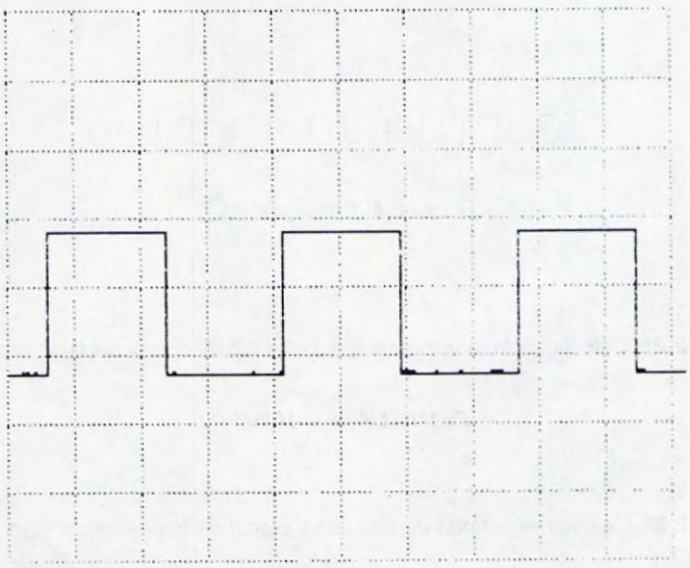


Figura 2.6. Cronogramas en un analizador lógico

2.11. INTRODUCCION AL ANALISIS DE SEÑALES

A continuación se realiza un breve estudio teórico de las señales básicas ideales más frecuentes en el análisis de circuitos. Las señales parábola y rampa de las que proceden los escalones y pulsos no son de aplicación a los circuitos básicos de puertas lógicas que constituyen el contenido de este libro por lo que se prescinde de su inclusión.

2.11.1. Señal impulso

Es una señal de mínima duración y máximo valor, que tiene como peculiaridad que el área del rectángulo infinitesimal que determina es igual a la unidad. Si el área es la expresión de una integral, su ecuación analítica sería la siguiente:

$$A=\int_{-\infty}^{\infty} f(t)dt=\int_{-\infty}^{-a} f(t)dt+\int_{-a}^{a} f(t)dt+\int_{a}^{\infty} f(t)dt \qquad [2.6]$$

La primera y tercera integrales son nulas para el impulso, al no existir la función en esos intervalos. Si f(t) se sustituye por la función delta de Dirac y los límites de la integral se hacen infinitésimos, cuando tienda a cero disminuirá el ancho del rectángulo, aumentando la altura del mismo, manteniéndose el área constante e igual a la unidad. Su figura representativa es la 2.7. La expresión [2.6] quedará de la siguiente forma:

$$A_i=\int_{-\infty}^{\infty} f(t)dt=\int_{-e/2}^{e/2} \delta(t)dt=\int_{-e/2}^{e/2} 1/\epsilon dt=1 \qquad [2.7]$$

2.11.2. Señal pulso

Es una señal de duración finita y valores constantes, delimitada entre dos instantes de tiempo t_1 y t_2. Se define como diferencia entre dos escalones generados en t_1 y t_2 respectivamente. El límite inferior del pulso cuando el tiempo de actuación se reduce al mínimo es un impulso. Su expresión representativa se indica mediante [2.8] y la figura 2.8 muestra la señal genérica.

$$A_p=\int_{-\infty}^{\infty} f(t)dt=\int_{t_1}^{t_2} Cdt=C(t_2-t_1) \qquad [2.8]$$

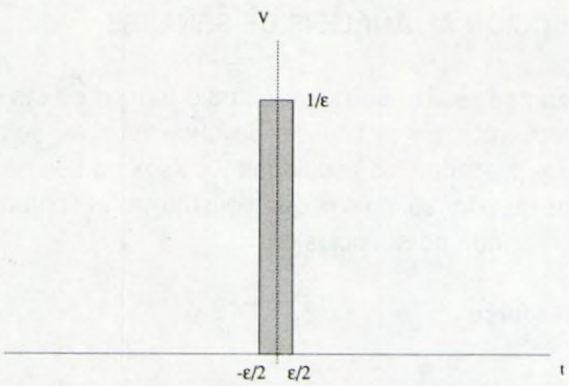


Figura 2.7. Señal impulso

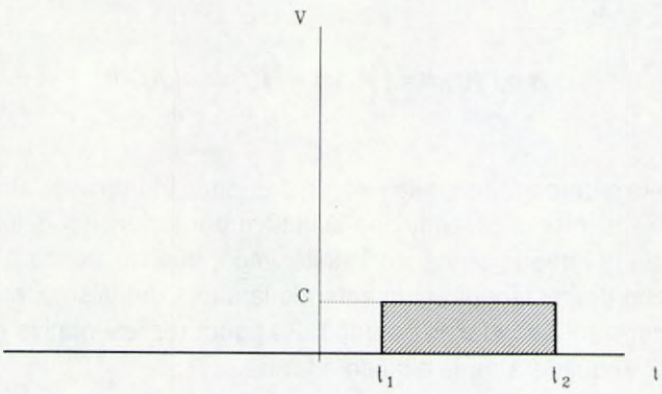


Figura 2.8. Señal pulso

2.11.3. **Señal escalón**

Es una señal de duración infinita, con valores constantes, delimitada entre un instante de tiempo t_1 y el infinito. Su integral representativa está indicada mediante [2.9] y su figura genérica la 2.9. El área definida por la integral es infinita y su desplazamiento en el tiempo puede permitir la obtención de pulsos mediante diferencia de señales según se obtiene de [2.10] y la figura 2.10.

$$A_e = \int_{-\infty}^{\infty} f(t)dt = \int_{t_1}^{\infty} C dt = \infty \qquad [2.9]$$

$$A_p = A_e^1 - A_e^2 = \int_{t_1}^{\infty} C dt - \int_{t_2}^{\infty} C dt = \int_{t_1}^{t_2} C dt = C(t_2 - t_1) \qquad [2.10]$$

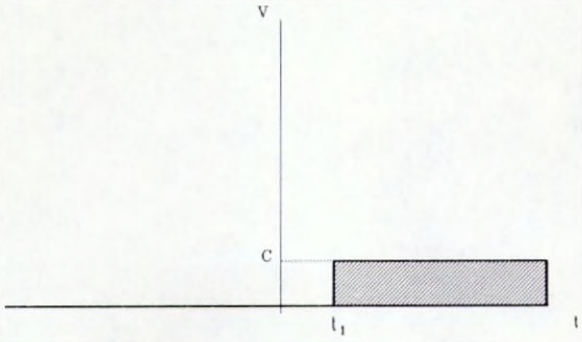


Figura 2.9. Señal escalón

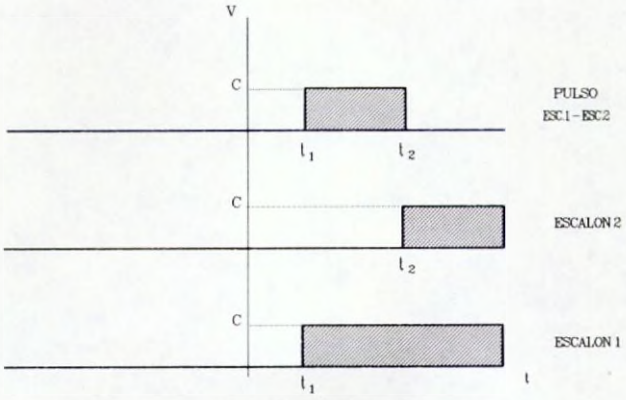


Figura 2.10. Diferencia de escalones

TEMA 3
TRANSMISION DE LA INFORMACION

3

TRANSMISION DE LA INFORMACION

3.1. INTRODUCCION

Después de analizar en el tema anterior el sistema de base dos o binario, se verá en este tema el almacenamiento y la transmisión de información. No sólo se tratará el aspecto puramente teórico y de desarrollo de los códigos, sino que se analizarán asimismo sus aplicaciones prácticas. Se determinarán las técnicas de transmisión de datos así como los bloques genéricos que intervienen en el diseño de los circuitos que realizan dichas funciones.

El análisis en coma flotante se realiza para los procedimientos más conocidos, existiendo múltiples variantes en los euipos informáticos. Se introducirá el concepto de paridad y su aplicación a la detección y corrección de errores en transmisión. Como apéndice final se realiza una breve introducción a los códigos ASCII, EBCDIC, Telegráfico, etc., que constituyen la base de codificación de cifras, letras y símbolos en equipos informáticos y de telecomunicación.

3.2. REPRESENTACION EN COMA FIJA

En la representación visual de un número en cualquier sistema de representación, distinguiremos una parte entera, otra fraccionaria y una separación entre ellas mediante coma. La transmisión y utilización de señales binarias es el equivalente que por comodidad utilizamos para manipular señales eléctricas del interior de los equipos digitales.

Sin embargo, cuando se representan números binarios existe la disyuntiva de la colocación de la coma de separación de las partes entera y fraccionaria, y de su codificación para poder ser almacenada. Analizando por ejemplo un registro de almacenamiento de información de 8 bits, se puede descomponer en dos

bloques de 4 bits, equivalente cada uno a un dígito del sistema de representación hexadecimal. Como se puede apreciar en el esquema representativo de la figura 3.1, no existe sitio físico para la coma de separación de ambas partes, por lo que al no representarse dicho elemento es necesario determinar unos criterios que indiquen al sistema la existencia de la coma y su situación.

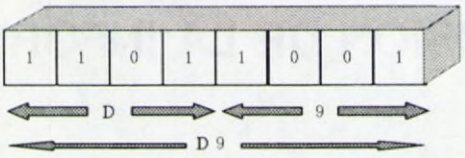


Figura 3.1. Registro de 8 bits

Se denomina representación en coma fija al sistema que bien por medio del usuario o por previa disposición del equipo, establece un número fijo de dígitos fraccionarios para las operaciones que se hayan de realizar. La coma por tanto no se almacena como información, pero el sistema conoce exactamente los bits que pertenecen a la parte entera y los que pertenecen a la parte fraccionaria. (El hecho de que la coma numérica no se almacene no quiere decir que en la escritura de textos por ejemplo, la coma empleada como signo de puntuación no haya de ser codificada y almacenada como un signo de puntuación).

En algunos equipos se pueden seleccionar tanto los dígitos de la parte fraccionaria como el posible redondeo de las últimas cifras, mientras que en otros no se puede realizar tal operación por parte del usuario sino que es el propio equipo el que fija dicha situación al haber sido previamente delimitado en su fabricación. Reglas básicas de representación en coma fija:

* Si cualquiera de las partes entera o fraccionaria tiene un número de bits menor que las cifras fijadas para las mismas, se completarán los huecos con "ceros". (En la realización física de esta operación los huecos no se "rellenan con ceros", sino que es la forma léxica de expresar dicha situación. Cuando una célula elemental de un registro no es activada con una señal "1", se supone descargada o inactiva, lo que equivale a tener el mínimo de tensión en sus terminales, y por tanto estamos ante el "0" lógico, valor que siempre está presente en cualquier elemento del sistema cuando no existe activación del mismo.)
* Si cualquiera de las partes entera o fraccionaria tiene un número de bits mayor que las cifras fijadas para las mismas, se truncarán los datos que

excedan de dichos valores, perdiéndose por tanto la información que pretendían transmitir. El truncamiento se puede producir tanto en la parte entera como en la fraccionaria.

Este sistema de representación tiene el peligro de la pérdida de información dependiendo del número de cifras fraccionarias que se establezcan. Por ello es necesario constituir otro sistema que evite este problema y que permita una mayor precisión en las operaciones que realice, así como mayor capacidad de almacenamiento. Es importante hacer notar que el primer bit de las cifras que se transmitan puede ser considerado como bit de signo, teniendo entonces en cuenta el criterio de asignación "0" para positivos y "1" para negativos, y disminuyendo en un bit el número de términos destinados a las partes entera y fraccionaria.

El procedimiento de **truncamiento** de la información es el más simple de los utilizados por los equipos digitales. Existen asimismo las posibilidades de obtener un resultado por **redondeo**, tomando el valor más cercano al correspondiente al bit menos significativo, y por **aproximación forzosa a 1**, que determina siempre que el bit menos significativo sea "1" en caso de exceso en el número de cifras.

*## **Ejemplo 3.1**: Se dispone de un registro de 8 bits y se determinan 2 bits para la parte fraccionaria. Almacenar el número decimal 13.*

Solución:

$13_{10}=1101_2$

Número almacenado: 00110100

*## **Ejemplo 3.2**: Se dispone de un registro de 8 bits y se determinan 2 bits para la parte fraccionaria. Almacenar el número decimal 13,8125.*

Solución:

$13{,}8125_{10}=1101{,}1101_2$

Número almacenado: 00110111

*## **Ejemplo 3.3**: Se dispone de un registro de 8 bits y se determinan 2 bits para la parte fraccionaria. Almacenar el número decimal 218,625.*

Solución:

$218{,}625_{10}=11011010{,}101_2$

Número almacenado: 01101010

*## **Ejemplo 3.4**: Se dispone de un registro de 8 bits y se determinan 2 bits para la parte fraccionaria, teniendo en cuenta además que se almacenan números con signo. Almacenar el número decimal -218,625.*

Solución:

$-218{,}625_{10} = 1\ 110011010{,}101_2$

Número almacenado: 11101010

3.3. REPRESENTACION EN COMA FLOTANTE

El sistema de representación en coma flotante elimina los problemas indicados para la representación en coma fija, pero dependiendo de los equipos existen diversas posibilidades, formatos, limitaciones, ventajas e inconvenientes, mayor o menor utilización de los mismos, y limitaciones que imponen a la representación de información. Las posibilidades de truncamiento y redondeo son las mismas que en sistema de coma fija. Como introducción a este sistema se aplica el equivalente en sistema decimal, comunmente denominado notación científica.

$$3497=3497*10^0=349{,}7*10^1=3{,}497*10^3=0{,}3497*10^4=34970*10^{-1}$$

Estas expresiones tienen unos términos comunes, que se denominan signo, mantisa y exponente, agrupándose de la siguiente forma:

$$N=(s)m*B^e \qquad [3.1]$$

N=Número en un sistema de base B
s=signo
m=mantisa
e=exponente
B=base del sistema de numeración

Aplicando la expresión [3.1] al sistema binario se obtiene la expresión básica de la que se obtiene la representación en coma flotante:

$$N=(s)m*2^e \qquad [3.2]$$

El desarrollo de la expresión binaria [3.2] conduce a diferentes posibilidades de representación, igual que en el formato científico decimal (La mantisa está codificada en binario y el exponente en decimal para mayor claridad):

$$-1100{,}11 = -1100{,}11 * 2^{0} = -110011 * 2^{-2} = 0{,}110011 * 2^{4}$$

3.3.1. **Formato para números enteros**

Este formato prácticamente es de escasa utilización en los actuales equipos, pero aún existen líneas de fabricación que consideran tal posibilidad. El orden de representación será **e/s/m**, aunque a veces se expresa como **e/m**, considerando el bit de signo de la mantisa como parte de dicho bloque.

Dado que pueden existir números con exponente positivo o negativo, se establece el criterio de representación en EXCESO para el exponente, con lo cuál se evita el análisis del bit de signo del mismo. El exponente se representa en exceso 2^{n-1}, siendo n el número de bits que se destinan al mismo. Este concepto de exceso quiere decir que dicho valor se añadirá al exponente obtenido en cada caso, produciéndose una suma si es positivo o una resta si es negativo, pero obteniéndose como resultado siempre un número mayor o igual que cero. Al tratarse de números enteros, si la mantisa del número que se procese tiene un número de cifras menor que las determinadas, se completarán los huecos con "ceros", colocándose a la izquierda de los valores utilizados.

La utilización de números con partes entera y fraccionaria conlleva la previa conversión de los mismos a parte entera únicamente, con la consiguiente aparición del producto de potencia de la base correspondiente. El rango de representación de los exponentes estará comprendido entre los siguientes valores:

Valor máximo: $2^{n-1}-1$

Valor mínimo: -2^{n-1}

A veces el valor mínimo se representa añadiéndole una unidad al indicado, puesto que daría cero como resultado del exceso y no se consideraría como exponente un resultado nulo.

3.3.2. **Formato no estandar para números fraccionarios**

Este procedimiento es similar al empleado para números enteros, pero teniendo en cuenta que sólo se representarán números con parte fraccionaria, y por tanto la parte entera de los números que se procesen habrá de ser convertida a parte fraccionaria. El procedimiento de números fraccionarios es

el más utilizado en los equipos digitales, aunque existen diversas posibilidades que se analizarán a continuación.

En este tipo de representación se introduce el concepto denominado **NORMALIZACION**, que consiste en transformar cualquier número a una cifra con parte entera cero y parte fraccionaria que cumpla que su primer bit, el situado a continuación de la coma, sea una cifra significativa, o lo que es lo mismo, un "1". Si se representa un número en formato no normalizado, el primer bit significativo puede ocupar cualquier posición, con lo que se podrán obtener soluciones diferentes, mientras que en formato normalizado la solución es única.

También se introduce el concepto de **BIT IMPLICITO** en algunos equipos, y se basa en no utilizar en el almacenamiento de datos el primer bit de la mantisa normalizada, que siempre será un "1", con lo que se consigue mayor rango de almacenamiento. El exceso para representar el exponente tiene la misma expresión que en el formato para números enteros, es decir, 2^{n-1}.

La parte correspondiente a la mantisa, estará formada por los bits situados a partir de la coma y los huecos existentes se completarán con "ceros" a la derecha de la última cifra de la mantisa que se utilice. El orden de representación de los términos será **e/s/m** igual que en el formato anterior, ó **e/m** si se incluye en m el bit de signo. Este orden de representación no es demasiado utilizado, denominándose formato NO ESTANDARIZADO, existiendo otras dos posibilidades de representación más utilizadas que se analizarán en los siguientes apartados. El rango de representación de los exponentes será el mismo que se definió en el apartado anterior para los números enteros.

Es interesante comprobar los datos ofrecidos por algunos fabricantes de equipos informáticos. Así por ejemplo, algunos equipos VAX tienen establecido el formato de 8 bits para exponente y 25 bits para mantisa, teniendo en cuenta que incluye bit de signo y utiliza bit implícito, contabilizado en el total de bits de la mantisa, disponiendo realmente de 24 posiciones de almacenamiento de información. La representación del exponente se realizará en exceso 128, teniendo en cuenta la fórmula de cómputo del mismo. Algunos equipos IBM disponen de 7 bits de exponente y 25 de mantisa, que incluye bit de signo pero no utiliza bit implícito. La representación del exponente se realizará por tanto en exceso 64.

3.3.3. **Formato estandar para números fraccionarios**

Es el formato de representación más utilizado, y corresponde a la referencia IEEE 754 (Institute for Electric and Electronic Engineers). Su representación exige exclusivamente el empleo de números con parte fraccionaria, y el orden de sus términos será el de **s/e/m** .

Utiliza términos **NORMALIZADOS** y **BIT IMPLICITO**, aspectos descritos en el formato anterior. El exponente se representa en exceso $2^{n-1}-1$, teniendo n el mismo significado que en los otros formatos, es decir, el número de bits que ocupará el exponente. Existen dos versiones de este formato, que se denominan de **SIMPLE PRECISION** y de **DOBLE PRECISION** según el número de bits que se utilizan para dicha representación. Simple precisión es la denominación del bloque de representación de 32 bits, desglosado en 8 bits de exponente, 1 bit de signo y 24 bits de mantisa, incluyendo aquí el bit implícito. El exceso del exponente será de 127, y el rango de representación del mismo oscilará entre -127 y +128, teniendo en cuenta que al alcanzarse el valor máximo de +128 el sistema interpretará resultados especiales.

Doble precisión es la denominación del bloque de 64 bits, compuesto por 11 bits de exponente, 1 bit de signo y 53 bits de mantisa, incluyendo el bit implícito. El exceso del exponente será de 1023, y el rango de representación oscilará entre -1023 y +1024, con las salvedades a que dan lugar algunas combinaciones de los valores. Así por ejemplo, teniendo en cuenta ambas precisiones, las peculiaridades del valor final del exponente y de su mantisa son las siguientes:

* Si el exponente es 255 ó 2047 y la mantisa es distinta de cero, se considera como representación la inespecificación 0/0.

* Si el exponente es 255 ó 2047 y la mantisa es cero, se considera para bit de signo "0" como la representación de +Infinito, y para bit de signo "1" como -Infinito.

* Si el exponente es cero y la mantisa es distinta de cero, se representan números demasiado pequeños no normalizados.

* Si el exponente es cero y la mantisa es cero, se representa el número cero.

3.3.4. **Formato directo para números fraccionarios**

Está constituido de idéntica forma al formato estandar, pero presenta inversión en el orden de colocación de los términos, representándose como **s/m/e**. Todas las reglas de dicho formato le son de aplicación en cuanto a la normalización, exceso del exponente y precisiones, menos el bit implícito que no es de obligado cumplimiento, pudiéndose representar utilizando o no dicho bit.

*## **Ejemplo 3.5**: Representar en coma flotante para números enteros el número decimal -21,75 utilizando un registro de 16 bits con 6 bits de exponente y 10 bits de mantisa (incluyendo el signo).*

Solución:

$-21{,}75_{10}=-10101{,}11_2=-1010111_2*2^2$

$s=1$
$m=1010111$
$e=-2_{10}=-10_2$

Exceso del exponente: $2^5=32_{10}=100000_2$
Exponente final: 100000-000010=011110
Representación en coma flotante: 011110 1 001010111

*## **Ejemplo 3.6**: Representar en coma flotante para números enteros el número binario +10101110000 utilizando un registro de 16 bits con 5 bits de exponente y 11 bits de mantisa (incluyendo el signo).*

Solución:

$+10101110000_2=+1010111_2*2^4$

$s=0$
$m=1010111$
$e=4_{10}=100_2$

Exceso del exponente: $2^4=16_{10}=10000_2$
Exponente final: 10000+00100=10100
Representación en coma flotante: 10100 0 0001010111

*## **Ejemplo 3.7**: Representar en coma flotante para números fraccionarios con formato no estandarizado el número decimal -21,75 utilizando un registro de 16 bits con 6 bits de exponente y 10 bits de mantisa (incluyendo el signo).*

Solución:

$-21{,}75_{10}=-10101{,}11_2=-0{,}1010111_2*2^5$

$s=1$
$m=1010111$
$e=5_{10}=101_2$

Exceso del exponente: $2^5=32_{10}=100000_2$
Exponente final: 100000+000101=100101
Representación final sin usar bit implícito: 100101 1 101011100
Representación final usando bit implicito: 100101 1 010111000

*## **Ejemplo 3.8**: Representar en coma flotante para números fraccionarios con formato no estandarizado el número binario +0,001010111 utilizando un registro de 16 bits con 5 bits de exponente y 11 bits de mantisa (incluyendo el signo).*

Solución:

$+0,001010111_2=+0,1010111_2*2^{-2}$

s=0
m=1010111
$e=-2_{10}=-10_2$

Exceso del exponente: $2^4=16_{10}=10000_2$
Exponente final: 10000-00010=01110
Representación final sin usar bit implícito: 01110 0 1010111000
Representación final usando bit implicito: 01110 0 0101110000

*## **Ejemplo 3.9**: Representar en coma flotante para números fraccionarios con formato estandar el número decimal -25,25 utilizando un registro de simple precisión.*

Solución:

$-25,25_{10}=-11001,01_2=-0,1100101_2*2^5$

s=1
m=1100101
$e=5_{10}=101_2$

Exceso del exponente: $2^7-1=127_{10}=1111111_2$
Exponente final: 1111111+00000101=10000100
Representación final: 1 10000100 01011100000000000000000

*## **Ejemplo 3.10**: Representar en coma flotante para números fraccionarios con formato estandar el número binario +0,000000001011111000111 1 utilizando un registro de simple precisión.*

Solución:

$+0,000000001011111000111_{10}=+0,10111110001111_2*2^{-8}$

s=0
m=10111110001111
$e=-8_{10}=-1000_2$

Exceso del exponente: $2^7-1=127_{10}=1111111_2$
Exponente final: 1111111-00001000=01110111
Representación final: 0 01110111 01011111000111100000000

*## **Ejemplo 3.11**: Representar en coma flotante formato directo el número decimal 104,5 utilizando un registro de simple precisión.*

Solución:
$104{,}5_{10}=110100{,}1_2=0{,}1101001_2*2^6$

s=0
m=1101001
$e=6_{10}=110_2$

Exceso del exponente: $2^7-1=127_{10}=1111111_2$
Exponente final: 1111111+00000110=10000101
Representación final sin usar bit implícito: 0 10000101 11010010000000000000000
Representación final usando bit implícito: 0 10000101 10100100000000000000000

*## **Ejemplo 3.12**: Representar en coma flotante formato directo el número decimal -6140 utilizando un registro de simple precisión.*

Solución:
$-6140_{10}=-1011111111100_2=-0{,}10111111111_2*2^{13}$

s=1
m=10111111111
$e=13_{10}=1101_2$

Exceso del exponente: $2^7-1=127_{10}=1111111_2$
Exponente final: 1111111+00001101=10001100
Representación final sin usar bit implícito: 0 10001100 10111111111000000000000
Representación final usando bit implícito: 0 10001100 01111111110000000000000

3.4. DETECCION Y CORRECCION DE ERRORES

En la transmisión de información, bien entre elementos de un mismo sistema digital o con elementos externos al mismo, se pueden producir fallos que modifiquen algún dígito y como consecuencia, produzcan un funcionamiento no deseado de dicho equipo. Para evitarlo, se han determinado unos sistemas de detección y de corrección de los errores que se analizan en el presente apartado. Dichos sistemas están basados en la generación de errores de **UN SOLO BIT**, causa más frecuente de los fallos de transmisión, siendo muy poco usuales los fallos de más de un bit de transmisión.

Cuando se producen dos errores, al no ocurrir ambos en el mismo instante de tiempo, el sistema casi en el cien por cien de los casos detecta y corrige ambos errores por separado. Sin embargo si la base de tiempos que controla un equipo digital no fuera tan perfecta como para separar ambos instantes de tiempo, se deben analizar dichos fallos mediante teoría de residuos. (Ver tratados especializados de álgebra y cálculo).

3.4.1. Bit de paridad

Se denomina de esta forma a un bit que se añade al conjunto de bits de información que se han de transmitir, teniendo en cuenta que el conjunto de dígitos significativos o "unos" del nuevo número transmitido (incluyendo el bit de paridad) debe cumplir con la siguiente regla según la paridad que se utilice:

PARIDAD PAR: El número total de bits "1" es **PAR**. (o en su defecto no existe ningún "1")

PARIDAD IMPAR: El número total de bits "1" es **IMPAR**.

El bit de paridad simplifica enormemente la detección de fallos, dado que no es necesario tener en cuenta ningún tipo de combinaciones de los elementos que forman la transmisión, sino sólo el número de unos que la constituyen. En caso de producirse un error en la transmisión de dicha información, el número de unos dejará de coincidir con la paridad, con lo cuál se detecta que se ha producido un error. Mediante comparación del número original con el transmitido erróneamente, se determina el dígito incorrecto, subsanándose el error con posterioridad. El esquema bloque que representa la generación del bit de paridad y su análisis se representa en la figura 3.2. En el tema 7 se

desarrolla el diseño de los circuitos correspondientes a la generación y a la detección de la paridad.

Analíticamente se empleará el criterio de colocar el bit de paridad en la posición más significativa. Este tipo de control detecta la presencia de un error pero no lo corrige. Se aplica principalmente en la transmisión de información entre ordenadores personales de 8 bits, utilizando la codificación de 7 bits del sistema ASCII más el octavo bit del control de paridad. Este octavo bit puede almacenarse para una posterior e inmediata transmisión o bien eliminarse y almacenar únicamente los bits correspondientes a la información básica transmitida.

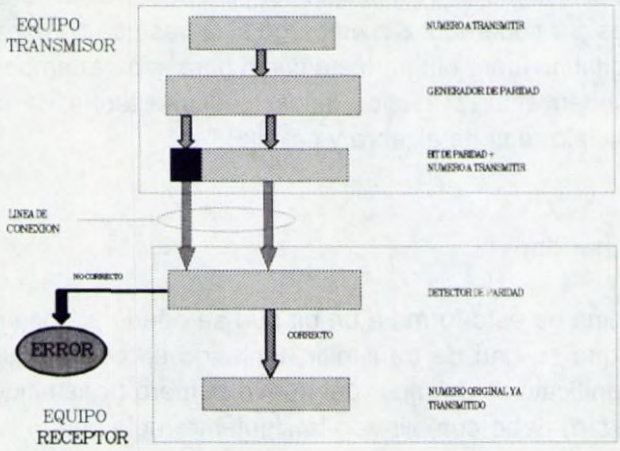


Figura 3.2. Bit de paridad

*## **Ejemplo 3.13**: Transmitir con paridad par el número decimal 71.*

Solución:

$$71_{10}=1000111_2$$

Número total de unos: 4
Bit de paridad PAR: 0
Resultado final:

01000111

*## **Ejemplo 3.14**: Transmitir con paridad impar el número decimal 72.*

Solución:

$$72_{10}=1001000_2$$

Número total de unos: 2
Bit de paridad IMPAR: 1
Resultado final:

11001000

3.4.2. **Corrección de errores por fila y columna**

Este sistema detecta y corrige los fallos de transmisión ya que localiza no sólo el conjunto de bits erróneos sino la posición exacta del error. Para ello, se realiza un doble control de paridad tanto en cada una de las filas como en las columnas, analizando la totalidad de los bits transmitidos. Eso significa tener una columna para el bit de paridad de fila horizontal, que se designará por BPF, y es necesario crear una nueva fila que corresponderá al bit de paridad de todas las columnas, que se designará por BPC. Se realiza este control cuando se trata de transmitir un conjunto o bloque de registros de información y se desea realizar la corrección de posibles errores sobre los mismos. El control se aplica a cada bloque de datos, perdiéndose la información de BPC generada para el bloque anterior.

El bit de paridad de cada fila se ha ido generando a medida que se creaba la fila correspondiente y puede perfectamente encontrarse almacenado en una memoria junto con los otros bits, mientras que los bits de paridad de cada columna se determinan en el momento en que se transmite el conjunto de información, generándose únicamente para dicho bloque y no almacenándose para posteriores transmisiones de otros bloques. La nueva fila que se obtiene como consecuencia de la generación de los bits de paridad de las columnas no lleva análisis de bit de paridad de fila si se utilizan ambas paridades par e impar, mientras que a la columna BPF, por encontrarse ya creada sí se le realiza análisis de paridad de columna. El esquema bloque que representa la transmisión por fila y columna se representa en la figura 3.3.

Si se produce un error, se generará una nueva tabla de transmisión conteniendo dicho bit erróneo, que debe ser detectado y corregido. El análisis de

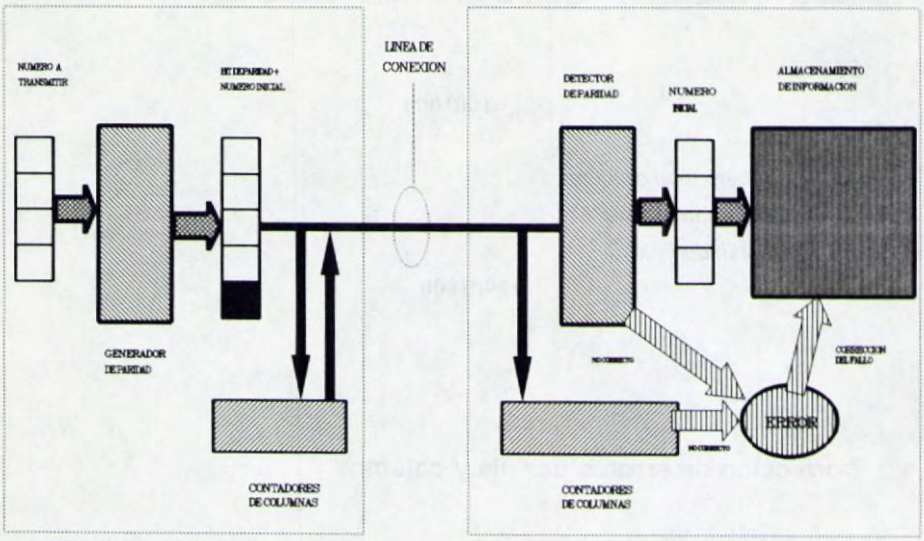


Figura 3.3. Paridad por fila y columna

las paridades horizontales y verticales del bloque detectará una fila y una columna que no cumplen la paridad, y el punto de corte de ambas localizará el bit erróneo que será corregido mediante complementación.

***## Ejemplo 3.15**: Determinar los bits de paridad necesarios para realizar un control de errores por fila y columna empleando paridad impar, si se desean transmitir los números decimales 102, 86, 89, 22 y 79.*

Solución:

BPF
1 1 1 0 0 1 1 0
1 1 0 1 0 1 1 0
1 1 0 1 1 0 0 1
0 0 0 1 0 1 1 0
0 1 0 0 1 1 1 1

BPC 0 1 0 0 1 1 1 1

*## **Ejemplo 3.16**: Suponiendo que se han empleado los elementos del ejemplo 3.7, y se produce un fallo según la tabla que se reproduce a continuación, detectar y corregir el error.*

BPF
1 1 1 0 0 1 1 0
1 1 0 1 0 1 1 0
1 1 0 1 0 0 0 1
0 0 0 1 0 1 1 0
0 1 0 0 1 1 1 1

BPC 0 1 0 0 1 1 1 1

Solución:
El análisis de las paridades detecta incorrecciones en la tercera fila y en la quinta columna. El cruce de ambas incorrecciones, fila y columna, determina la posición del error transmitido, el cuál será corregido mediante complementación.

3.4.3. **Corrección de errores mediante códigos de Hamming**

Este método, detecta y corrige los fallos de un bit cometidos en la transmisión de información. Se basa en la introducción de un número determinado de bits de paridad en la cifra a transmitir, construidos de acuerdo con unas determinadas combinaciones de dígitos, que se detallan a continuación. De esta forma se consigue que al transmitir un número con dichos códigos correctores, si existe un fallo de transmisión, se detecta no sólo el fallo existente sino la posición que ocupa, corrigiéndose entonces mediante complementación.

Si denominamos con "n" al número de bits de paridad que se deben introducir para detectar y corregir los fallos, y con "m" al número de bits que tendrá la cifra que se quiere transmitir, se deberá cumplir la siguiente relación, que permite obtener el valor "n":

$$2^n \geq n+m+1 \qquad [3.3]$$

La cifra transmitida con los códigos correctores tendrá n+m bits. Los "n" bits de paridad ocuparán las posiciones que determinan la siguiente expresión de potencias, tomando "p" valores enteros:

$$2^p \qquad [3.4]$$

$$0 \leq p \leq n-1$$

Para no desvirtuar el orden de expresión de los números a transmitir, se establece el orden de colocación de los bits de **IZQUIERDA A DERECHA**, asignando la **POSICION 1** al bit situado más a la izquierda (No existe posición cero). Los valores correspondientes a los bits de paridad se obtienen mediante combinaciones de los bits que forman el número a transmitir. El método más sencillo para obtener dichas combinaciones tiene como base la tabla de números binarios que se forma con las combinaciones de varios bits, por ejemplo la tabla 3.1 construida para 4 bits. Para mayor claridad, se expresa dicha tabla en forma horizontal en vez de vertical, para poder asemejarla a la relación de posiciones que se ha descrito anteriormente.

(Pos.)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
C_1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
C_2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
C_3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
C_4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Tabla 3.1. Combinaciones de Hamming

A la primera línea de dígitos binarios se le asigna C_1, a la segunda línea C_2, a la tercera C_3, a la cuarta C_4 y así sucesivamente si se amplía la tabla con más bits. En cada línea se tendrán en cuenta las posiciones que poseen un "1", y se definirán las combinaciones necesarias para obtener los bits de paridad y para la posterior detección de los errores. Los coeficientes y combinaciones que se obtienen son por tanto los siguientes para la tabla 3.1 formada por 4 bits:

$$C_1=1+3+5+7+9+11+... \qquad [3.5]$$

$$C_2=2+3+6+7+10+11+... \qquad [3.6]$$

$$C_3=4+5+6+7+12+13+... \qquad [3.7]$$

$$C_4=8+9+10+11+12+13+... \qquad [3.8]$$

Dependiendo del tipo de paridad que se emplee, par o impar, y de acuerdo con las reglas de dichos tipos de paridades, se irán asignando valores a los bits correspondientes. Una vez generados los bits de paridad y transmitido el número completo, el análisis de los fallos de transmisión se realiza de la siguiente forma:

* Se forma el número binario:

$$C_n+C_{n-1}+...+C_3+C_2+C_1=C \qquad [3.9]$$

* Se analizan las combinaciones obtenidas C_1, C_2, C_3, etc., utilizando las mismas combinaciones que en la obtención de los bits de paridad

* Si la suma de "unos" transmitidos en el análisis de cada combinación C_i corresponde con la paridad que se ha establecido, se asignará un "0" a dicha combinación en el número binario C

* Si la suma de "unos" no corresponde con la paridad establecida se le asignará un "1" a dicha combinación en el número C

* El número C determinará la posición del bit erróneo.

* Complementando el valor de dicho bit erróneo se subsana el error cometido en la transmisión.

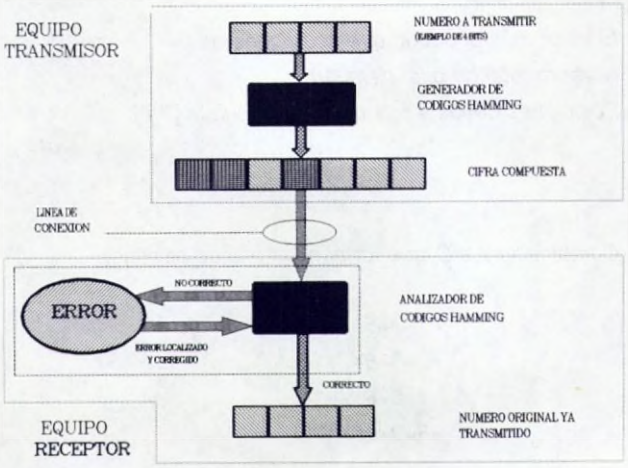


Figura 3.4. Transmisión empleando códigos de Hamming

Se puede diseñar un procedimiento con estructura similar a la matricial para analizar los errores producidos en la transmisión por Hamming. La ecuación genérica definida es la siguiente:

[P]=[N]*[B]

La matriz [N] es una matriz fila que corresponde al número completo transmitido (incluyendo códigos de paridad). La matriz [B] es la matriz de orden (n+m)xn basada en la tabla binaria, con "(n+m)" filas correspondientes al número de bits a transmitir y "n" columnas correspondientes al número de bits de paridad empleados. Haciéndola equivalente a la tabla 3.1, "(n+m)" se corresponde con las posiciones y "n" con los coeficientes C_i. El "producto" de ambas matrices produce unos números que pueden o no coincidir con la paridad establecida. Si coincide se le asigna "0" al resultado y en caso contrario "1". Si la matriz [P] es [0 0 .. 0] indica que no existe error. Cualquier otro valor indicará directamente en binario la posición del error cometido.

*## **Ejemplo 3.17**: Transmitir empleando códigos de Hamming con paridad par el número binario 110001.*

Solución:

Número de bits a transmitir: m=6
Sustituyendo en [3.3] se obtiene:

$$2^n \geq n+7$$

El valor más pequeño que se obtiene es n=4
Número total de bits: n+m=10
Posiciones de los 4 bits de paridad según [3.4]:

$$2^0,2^1,2^2,2^3=1,2,4,8$$

Combinaciones C_i que forman los bits de paridad:

C_1=1+3+5+7+9="0"
C_2=2+3+6+7+10="0"
C_3=4+5+6+7="1"
C_4=8+9+10="1"

El número a transmitir es: 0011100101

*## **Ejemplo 3.18**: Transmitir empleando códigos de Hamming con paridad par el número binario 11.*

Solución:

Número de bits a transmitir: m=2

Sustituyendo en [3.3] se obtiene:

$$2^n \geq n+3$$

El valor más pequeño que se obtiene es n=3

Número total de bits: n+m=5

Posiciones de los 3 bits de paridad según [3.4]:

$$2^0, 2^1, 2^2 = 1, 2, 4$$

Combinaciones C_i que forman los bits de paridad:

C_1=1+3+5="0"

C_2=2+3="1"

C_3=4+5="1"

El número a transmitir es: 01111

*## **Ejemplo 3.19**: Al transmitir un número empleando códigos de Hamming con paridad impar se ha cometido un error. Si la transmisión recibida ha sido 1011100001, obtener la posición del fallo para que pueda ser complementado.*

Solución:

Número total de bits: n+m=10

Número de bits de paridad: n=4

Combinaciones C_i que forman el número C:

C_1=1+3+5+7+9="0"

C_2=2+3+6+7+10="1"

C_3=4+5+6+7="1"

C_4=8+9+10="0"

Posición del error según [3.9]: $C=0110_2=6_{10}$

Número correcto: 1011110001

*## **Ejemplo 3.20**: Al transmitir un número empleando códigos de Hamming con paridad par se ha cometido un error. Si la transmisión recibida ha sido 001111, obtener la posición del fallo para que pueda ser complementado.*

Solución:

Número total de bits: n+m=6

Número de bits de paridad: n=3

Combinaciones C_i que forman el número C:

C_1=1+3+5="0"

C_2=2+3+6="0"

C_3=4+5+6="1"

Posición del error según [3.9]: $C=100_2=4_{10}$

Número correcto: 001011

*## **Ejemplo 3.21**: Comprobar matricialmente si existe error en la transmisión efectuada en el ejemplo 3.20.*

Solución:

$$[P]=[001111]\begin{bmatrix}0&0&1\\0&1&0\\0&1&1\\1&0&0\\1&0&1\\1&1&0\end{bmatrix}=[1\ \ 0\ \ 0]$$

Posición del error: $100_2=4_{10}$

Número correcto: 001011

*## **Ejemplo 3.22**: Comprobar matricialmente si existe error en la transmisión efectuada en el ejemplo 3.19.*

Solución:

$$[P]=[1011100001]\begin{bmatrix}0&0&0&1\\0&0&1&0\\0&0&1&1\\0&1&0&0\\0&1&0&1\\0&1&1&0\\0&1&1&1\\1&0&0&0\\1&0&0&1\\1&0&1&0\end{bmatrix}=[0\ \ 1\ \ 1\ \ 0]$$

Posición del error: $0110_2=6_{10}$
Número correcto: 1011110001

***## Ejemplo 3.23**: Comprobar matricialmente si existe error en una transmisión si se recibe el número 101011011 con paridad par.*

Solución:

$$[P]=[101011011]\begin{bmatrix}0&0&0&1\\0&0&1&0\\0&0&1&1\\0&1&0&0\\0&1&0&1\\0&1&1&0\\0&1&1&1\\1&0&0&0\\1&0&0&1\end{bmatrix}=[0\ \ 0\ \ 0\ \ 0]$$

Posición del error: 0000
El número es correcto. No hay error.

3.5. **CODIGOS ESPECIALES**

A continuación se detallan varios tipos especiales de códigos de representación, empleados fundamentalmente para transmisión de datos, aunque alguno de ellos posteriormente a su concepción original ha sido empleado como codificador de información interna de los equipos. Todos ellos se basan en el número posible de combinaciones que se pueden realizar con varios dígitos binarios, asimilando cada una de estas combinaciones a una letra del alfabeto, un número del sistema decimal o un signo de puntuación.

3.5.1. **El código ASCII**

ASCII son las siglas de American Standard Code for Information Interchange (Código estandar americano para intercambio de información). Se basa en las posibles combinaciones de 7 bits, lo que da lugar a 128 diferentes valores. Su uso principal es para transmisión de datos, acompañándose de un bit de paridad, aunque se usa ampliamente en la codificación interna de los teclados de ordenadores y en la conexión con sus periféricos. Se aplica a transmisión en cinta perforada y cinta magnética de 8 y 9 pistas, en las cuáles se definen códigos detectores de error. Su codificación responde a los primeros 128 códigos de la tabla 3.2, utilizándose normalmente los símbolos comprendidos entre el 32 y 126.

3.5.2. **El código EBCDIC**

Son las siglas de Extended Binary Coded Decimal Interchange (Sistema extendido decimal codificado en binario para intercambio de información). Se basa en las combinaciones de 8 bits, lo que permite hasta 256 códigos diferentes. Se denomina extendido o ampliado porque la mitad de los valores posibles se equiparan con los del código ASCII, estando los demás disponibles para códigos particulares de transmisión y escritura. Se suele emplear principalmente en los grandes equipos informáticos. Una variante de este sistema es el BCDIC o Binary Coded Decimal Interchange, constituido por 7 bits, similar al ASCII. Normalmente se le suele equiparar con el denominado ASCII-Extendido cuya codificación se expresa mediante la tabla 3.2.

00000000		00110100	4	01101000	h	10011100	£	11010000	╨
00000001	☺	00110101	5	01101001	i	10011101	¥	11010001	╤
00000010	☻	00110110	6	01101010	j	10011110	₧	11010010	╥
00000011	♥	00110111	7	01101011	k	10011111	ƒ	11010011	╙
00000100	♦	00111000	8	01101100	l	10100000	á	11010100	╘
00000101	♣	00111001	9	01101101	m	10100001	í	11010101	╒
00000110	♠	00111010	:	01101110	n	10100010	ó	11010110	╓
00000111	•	00111011	;	01101111	o	10100011	ú	11010111	╫
00001000	◘	00111100	<	01110000	p	10100100	ñ	11011000	╪
00001001	○	00111101	=	01110001	q	10100101	Ñ	11011001	┘
00001010	■	00111110	>	01110010	r	10100110	ª	11011010	┌
00001011	♂	00111111	?	01110011	s	10100111	º	11011011	█
00001100	♀	01000000	@	01110100	t	10101000	¿	11011100	▄
00001101	♪	01000001	A	01110101	u	10101001	⌐	11011101	▌
00001110	♫	01000010	B	01110110	v	10101010	¬	11011110	▐
00001111	☼	01000011	C	01110111	w	10101011	½	11011111	▀
00010000	►	01000100	D	01111000	x	10101100	¼	11100000	α
00010001	◄	01000101	E	01111001	y	10101101	¡	11100001	ß
00010010	↕	01000110	F	01111010	z	10101110	«	11100010	Γ
00010011	‼	01000111	G	01111011	{	10111111	»	11100011	π
00010100	¶	01001000	H	01111100	|	10110000	▒	11100100	Σ
00010101	§	01001001	I	01111101	}	10110001	▓	11100101	σ
00010110	▬	01001010	J	01111110	~	10110010	▓	11100110	µ
00010111	↨	01001011	K	01111111	⌂	10110011	▓	11100111	τ
00011000	↑	01001100	L	10000000	Ç	10110100	┤	11101000	Φ
00011001	↓	01001101	M	10000001	ü	10110101	╡	11101001	Θ
00011010	→	01001110	N	10000010	é	10110110	╢	11101010	Ω
00011011	←	01001111	O	10000011	â	10110111	╖	11101011	δ
00011100	∟	01010000	P	10000100	ä	10111000	╕	11101100	∞
00011101	↔	01010001	Q	10000101	à	10111001	╣	11101101	φ
00011110	▲	01010010	R	10000110	å	10111010	║	11101110	ε
00011111	▼	01010011	S	10000111	ç	10111011	╗	11101111	∩
00100000		01010100	T	10001000	ê	10111100	╝	11110000	≡
00100001	!	01010101	U	10001001	ë	10111101	╜	11110001	±
00100010	"	01010110	V	10001010	è	10111110	╛	11110010	≥
00100011	#	01010111	W	10001011	ï	10111111	┐	11110011	≤
00100100	$	01011000	X	10001100	î	11000000	└	11110100	⌠
00100101	%	01011001	Y	10001101	ì	11000001	┴	11110101	⌡
00100110	&	01011010	Z	10001110	Ä	11000010	┬	11110110	÷
00100111	'	01011011	[	10001111	Å	11000011	├	11110111	≈
00101000	(	01011100	\	10010000	É	11000100	─	11111000	°
00101001	)	01011101	]	10010001	æ	11000101	┼	11111001	∙
00101010	*	01011110	^	10010010	Æ	11000110	╞	11111010	·
00101011	+	01011111	_	10010011	ô	11000111	╟	11111011	√
00101100	,	01100000	`	10010100	ö	11001000	╚	11111100	ⁿ
00101101	-	01100001	a	10010101	ò	11001001	╔	11111101	²
00101110	.	01100010	b	10010110	û	11001010	╩	11111110	■
00101111	/	01100011	c	10010111	ù	11001011	╦	11111111	
00110000	0	01100100	d	10011000	ÿ	11001100	╠		
00110001	1	01100101	e	10011001	Ö	11001101	═		
00110010	2	01100110	f	10011010	Ü	11001110	╬		
00110011	3	01100111	g	10011011	¢	11001111	╧		

Tabla 3.2. Código ASCII-Extendido

3.5.3. **El código Hollerit**

Se basa en las posibles combinaciones de 12 bits, lo que permite 4096 diferentes valores. Dado que este código se emplea únicamente para tarjetas perforadas, hoy prácticamente en desuso, la gran cantidad de posibles combinaciones no tiene interés alguno, y sólo se emplean los necesarios para representar los mismos elementos que en el código ASCII. Actualmente se aplica minoritariamente en tarjetas magnéticas de calculadoras.

3.5.4. **El código telegráfico**

Se basa en las combinaciones de 5 bits, con 32 posibles valores. Como este número de combinaciones es demasiado pequeño para poder abarcar el alfabeto completo, los números del sistema decimal y algunos signos de puntuación, fue necesario dotar a este código de un sistema especial de autocodificación.

Para ello, de los 32 códigos, se emplean 6 para dígitos de control, y los 26 restantes tienen doble significado, como letra del alfabeto y como número del sistema decimal o signo de puntuación. En función de una manera peculiar de transmitir la información, se pueden interpretar las combinaciones con doble significado. Dos de los dígitos de control interpretan este doble significado, de tal forma que tras la aparición de uno de dichos dígitos, por ejemplo el que asigna las letras, todas las combinaciones serán tenidas en cuenta como letras, y si aparece el otro dígito, serán tenidas en cuenta como números o signos.

TEMA 4
ALGEBRA DE BOOLE

4

ALGEBRA DE BOOLE

4.1. INTRODUCCION

Se introduce en este tema el concepto de función, como agrupación de expresiones lógicas y representación de la lógica de los sistemas digitales. El álgebra de Boole es un procedimiento de representación de funciones basado en el sistema binario. Se desarrolla basándose en la teoría de conjuntos, desarrollada ampliamente en los tratados de álgebra moderna. Como el álgebra de Boole y el sistema binario de numeración son equivalentes en cuanto a que ambos sólo disponen de dos posibles valores, y ambos complementarios, la expresión y la representación de funciones booleanas será la equivalencia del sistema binario correspondiente.

Introduciendo inicialmente las expresiones y posibilidades del álgebra booleana, se pasará posteriormente a su equivalente en binario, y se definirán los elementos más empleados y representativos de la electrónica digital, las denominadas puertas lógicas. Estas, corresponden a las representaciones simbólicas de los circuitos electrónicos formados por semiconductores que realizan dichas operaciones. En el presente tema, se presentarán dichas funciones lógicas, dejando para el tema siguiente la operatividad y simplicidad de las mismas.

4.2. PROPIEDADES DEL ALGEBRA DE BOOLE

El álgebra de Boole está definido mediante expresiones, propiedades y funciones que determinan toda la moderna teoría matemática de conjuntos. Estas propiedades van a constituir la base para definir y simplificar las representaciones que se realicen en el sistema binario y en sus aplicaciones prácticas. Se definen como variables las expresiones algebraicas "A", "B", "C", etc.,

que pueden tomar los valores "0" y "1". La agrupación de dichas variables dará lugar a la aplicación de las siguientes propiedades booleanas para las operaciones de suma y producto (Aunque se indica en todas las operaciones para mayor claridad, puede omitirse el asterisco de la operación producto):

4.2.1. **Propiedad de identidad o idempotencia**

Operación de una variable consigo misma, que conduce a la misma variable como resultado.

$$A+A=A \qquad [4.1]$$

$$A*A=A \qquad [4.2]$$

4.2.2. **Propiedad conmutativa**

El orden de los términos en las operaciones de suma y producto no afecta al resultado final.

$$A+B=B+A \qquad [4.3]$$

$$A*B=B*A \qquad [4.4]$$

En caso de que las variables tomen valores constantes, las expresiones [4.3] y [4.4] conducen a los siguientes resultados:

$$A+0=0+A=A \qquad [4.5]$$

$$A*0=0*A=0 \qquad [4.6]$$

$$A+1=1+A=1 \qquad [4.7]$$

$$A*1=1*A=A \qquad [4.8]$$

$$0+0=0 \qquad [4.9]$$

$$1+0=1 \qquad [4.10]$$

$$1+1=1 \qquad [4.11]$$

$$0*0=0 \qquad [4.12]$$

$$0*1=0 \qquad [4.13]$$

$$1*1=1 \qquad [4.14]$$

4.2.3. **Propiedad de complementación**

$$\overline{A}=Complemento\ \ de\ \ A$$

$$\overline{A}+A=1 \qquad [4.15]$$

$$\overline{A}*A=0 \qquad [4.16]$$

4.2.4. **Propiedad de doble complementación o involución**

$$\overline{\overline{A}}=A \qquad [4.17]$$

4.2.5. **Propiedad asociativa**

Las variables se pueden agrupar de cualquier forma, utilizando además la propiedad de conmutatividad para intercambiar el orden y posición de los términos.

$$A+B+C=A+(B+C)=(A+B)+C=(A+C)+B \qquad [4.18]$$

$$A*B*C=A*(B*C)=(A*B)*C=(A*C)*B \qquad [4.19]$$

4.2.6. **Propiedad distributiva del producto respecto a la suma**

$$A+(B*C)=(A+B)*(A+C) \qquad [4.20]$$

Cuando una de las variables se vuelve a utilizar en otro término, se puede simplificar de la siguiente forma, constituyendo dos expresiones importantísimas en álgebra booleana:

$$A+(A*B)=(A+A)*(A+B)=A*(A+B)=A \qquad [4.21]$$

$$A+(\overline{A}*B)=(A+\overline{A})*(A+B)=1*(A+B)=A+B \qquad [4.22]$$

4.2.7. **Propiedad distributiva de la suma respecto al producto**

$$A*(B+C)=(A*B)+(A*C) \qquad [4.23]$$

Igual que en la propiedad distributiva de la suma, la utilización repetida de una variable conduce a las siguientes simplificaciones:

$$A*(A+B)=(A*A)+(A*B)=A+(A*B)=A \qquad [4.24]$$

$$A*(\overline{A}+B)=(A*\overline{A})+(A*B)=0+(A*B)=A*B \qquad [4.25]$$

4.2.8. **Teoremas de Morgan**

$$\overline{A+B}=\overline{A}*\overline{B} \qquad [4.26]$$

$$\overline{A*B}=\overline{A}+\overline{B} \qquad [4.27]$$

Son las dos propiedades más importantes y conocidas del álgebra booleana, que sirven además de base para la equivalencia de circuitos lógicos y sus posteriores aplicaciones prácticas. Se basan en la negación de funciones, utilizando las operaciones de suma y producto. Cada uno de los elementos o variables A y B puede representar cualquier función lógica, que a su vez, puede ser simplificable mediante cualquiera de las propiedades previamente descritas.

4.3. **FUNCIONES LOGICAS Y PUERTAS LOGICAS**

Basándose en las propiedades relacionadas anteriormente, se analizan las tres operaciones fundamentales que se han empleado, las de suma, producto y complemento. Se definen tablas, donde A y B representan las variables de entrada, expresándose cada una de las salidas mediante la operación lógica correspondiente F(A,B). Estas funciones, se representan mediante símbolos denominados puertas lógicas, que tal como se definieron anteriormente, constituyen la representación esquemática de los circuitos que realizan dichas operaciones.

Las expresiones de dichas puertas lógicas que se realizan a continuación, emplean la simbología más conocida y utilizada, la "American Standard", de difusión internacional y mayor aplicación que otras simbologías existentes. Se emplean los símbolos de las operaciones ya detalladas, así como los complementos de las mismas.

4.3.1. **Función lógica SUMA. Puerta lógica OR**

$$F(A,B)=A+B \qquad [4.28]$$

Se corresponde con los valores lógicos de la tabla 4.1 para la suma. Siempre genera salida "1" cuando al menos una de las entradas es "1". La

denominación de la puerta lógica es **OR**, y su esquema básico para 2 entradas se indica en la figura 4.1. Puede tener múltiples entradas.

A B	A+B
0 0	0
0 1	1
1 0	1
1 1	1

Tabla 4.1. Suma lógica

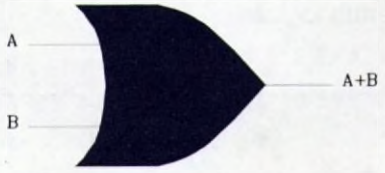


Figura 4.1. Puerta lógica OR

4.3.2. **Función lógica PRODUCTO. Puerta lógica AND**

$$F(A,B)=A*B \qquad [4.29]$$

Se corresponde con los valores lógicos de la tabla 4.2 para el producto. La salida será "1" únicamente cuando todas las entradas sean "1". La denominación de la puerta lógica es **AND**, y su esquema básico de 2 entradas se indica en la figura 4.2. Puede tener múltiples entradas.

A B	A*B
0 0	0
0 1	0
1 0	0
1 1	1

Tabla 4.2. Producto lógico

Figura 4.2. Puerta lógica AND

4.3.3. **Función lógica NEGACION. Puerta lógica NO**

$$F(A,B)=\overline{A} \qquad [4.30]$$

Corresponde a los valores expresados en la tabla 4.3 para la complementación. La salida siempre es el valor opuesto al de la entrada. La denominación de la puerta lógica correspondiente es **NO**, también llamada INVERSOR, y su esquema se representa en la figura 4.3. Se suele simbolizar sólo con el círculo cuando se emplea en las entradas o salidas de otras puertas lógicas.

A	$\overline{A}$
0	1
1	0

Tabla 4.3. Complementación lógica

Figura 4.3. Puerta lógica NO

4.3.4. **Función NEGACION DE LA SUMA. Puerta lógica NOR**

$$F(A,B)=\overline{A+B} \qquad [4.31]$$

Se corresponde con los valores lógicos de la tabla 4.4 para la suma complementada. Sólo genera salida "1" cuando todas las entradas son "0". La denominación de la puerta lógica es **NOR**, y su esquema básico para 2 entradas se indica en la figura 4.4. Puede tener múltiples entradas. Aplicando teoremas de Morgan se puede representar el esquema de la figura 4.4 mediante el montaje de la figura 4.5.

A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

Tabla 4.4. Negación de la suma lógica

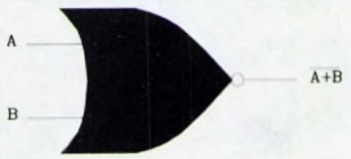


Figura 4.4. Puerta lógica NOR

Figura 4.5. Montaje equivalente a una puerta NOR

4.3.5. **Función NEGACION del PRODUCTO. Puerta lógica NAND**

$$F(A,B)=\overline{A*B} \qquad [4.32]$$

Se corresponde con los valores lógicos de la tabla 4.5 para el producto complementado. La salida será "0" únicamente cuando todas las entradas sean "1". La denominación de la puerta lógica es **NAND**, y su esquema básico para 2 entradas se indica en la figura 4.6. Puede tener múltiples entradas. Aplicando el teorema de Morgan correspondiente, la figura 4.6 puede representarse mediante el montaje de la figura 4.7.

A	B	$\overline{A*B}$
0	0	1
0	1	1
1	0	1
1	1	0

Tabla 4.5. Negación del producto lógico

Figura 4.6. Puerta lógica NAND

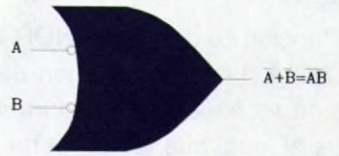


Figura 4.7. Esquema equivalente a una puerta NAND

4.3.6. **Función lógica OR-EXCLUSIVA. Puerta XOR**

$$F(A,B)=A\oplus B \qquad [4.33]$$

Se corresponde con los valores lógicos de la tabla 4.6 para la suma-exclusiva. Siempre genera salida "1" cuando las entradas toman valores diferentes. Se suele denominar función OR-EXCLUSIVA y también O-EXCLUSIVA. La puerta lógica es la **XOR** o **EXOR**, y su esquema se indica en la figura 4.8. Tiene normalmente sólo 2 entradas aunque hay montajes con múltiples entradas, formados por puertas básicas de 2 entradas digitales.

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	⊙

Tabla 4.6. O-Exclusiva

Figura 4.8. Puerta lógica XOR

4.3.7. **Función lógica Y-EXCLUSIVA. Puerta XNOR**

$$F(A,B)=A\odot B \qquad [4.34]$$

Se determina por los valores de la tabla 4.7 para el complemento de la suma-exclusiva, generando salida "1" cuando las entradas toman valores iguales.

La función se denomina NOR-EXCLUSIVA, Y-EXCLUSIVA o función **EQUIVALENCIA**. La denominación de la puerta lógica es **XNOR** o **EXNOR**, y su esquema se indica en la figura 4.9. Tiene sólo 2 entradas la puerta básica, aunque al igual que en la puerta XOR pueden existir montajes múltiples.

A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	1

Tabla 4.7. Y-Exclusiva

Figura 4.9. Puerta lógica XNOR

4.4. **EJEMPLOS**

*## **Ejemplo 4.1**: Demostrar que se cumple la igualdad [4.15] empleando la tabla de verdad.*

Solución:

A	B	AB	A+AB
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

*## **Ejemplo 4.2**: Simplificar mediante el álgebra de Boole la siguiente expresión algebraica:*

$$ABC+AB\overline{C}D+AB\overline{C}\overline{D}+\overline{A}+\overline{B}C$$

Solución:

$$BC+B\overline{C}D+BC\overline{D}+\overline{A}+\overline{B}C$$

$$C+B\overline{C}D+BC\overline{D}+\overline{A}$$

$$C+BD+\overline{A}$$

*## **Ejemplo 4.3**: Simplificar mediante el álgebra de Boole la siguiente expresión algebraica:*

$$(A+C)(\overline{BCD+\overline{A}C+\overline{A}B})$$

Solución:

$$(A+C)(\overline{BCD})(\overline{\overline{A}C})(\overline{\overline{A}B})$$

$$(A+C)(\overline{B}+\overline{C}+\overline{D})(A+\overline{C})(A+\overline{B})$$

$$A(\overline{B}+\overline{C}+\overline{D})(A+\overline{B})$$

$$A(\overline{B}+\overline{C}+\overline{D})$$

*## **Ejemplo 4.4**: Simplificar mediante el álgebra de Boole la siguiente expresión algebraica:*

$$\overline{\overline{D}\overline{C}\overline{B}A+\overline{D}CBA+DCB\overline{A}+D\overline{C}A\overline{B}+C\overline{B}+(A+\overline{C})}$$

Solución:

$$(\overline{D}\overline{C}BA+DCB\overline{A}+D\overline{C}A\overline{B})(\overline{C}+B)+A+\overline{C}$$

$$\overline{D}\overline{C}BA+DCB\overline{A}+A+\overline{C}$$

$$DB+A+\overline{C}$$

*## **Ejemplo 4.5**: Simplificar mediante el álgebra de Boole la siguiente expresión algebraica:*

$$\overline{\overline{A}\overline{B}\overline{C}D+(\overline{A+B+C+D})}+A\overline{B}\overline{C}\overline{D}+ABCD+\overline{A\overline{B}C(BCD+\overline{A}\overline{B}\overline{C})}+B$$

Solución:

$$\overline{\overline{A}\overline{B}\overline{C}D+(\overline{A+B+C+D})}+A\overline{B}\overline{C}\overline{D}+ABCD+\overline{A\overline{B}\overline{C}\overline{D}}+B$$

$$\overline{\overline{A}\overline{B}\overline{C}D+(\overline{A+B+C+D})}+\overline{A\overline{B}\overline{C}\overline{D}+1}+B$$

$$(A+\overline{B}+C+\overline{D})(A+B+C+D)+B$$

$$A+\overline{B}C+\overline{B}D+C+B\overline{D}+B$$

$$A+B+C+D$$

*## **Ejemplo 4.6**: Obtener el cronograma de salida de la puerta lógica OR de dos entradas cuyos valores están representados por los cronogramas A y B de la figura 4.10.*

Solución:

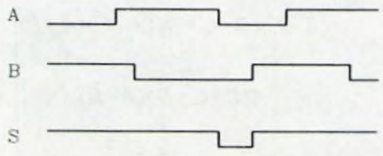


Figura 4.10

*## **Ejemplo 4.7**: Obtener el cronograma de salida de la puerta lógica AND de dos entradas cuyos valores están representados por los cronogramas A y B de la figura 4.11.*

Solución:

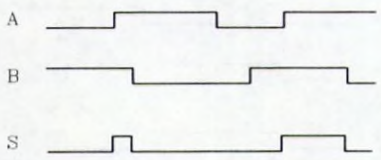


Figura 4.11

*## **Ejemplo 4.8**: Obtener el cronograma de salida de la puerta lógica NOR de dos entradas cuyos valores están representados por los cronogramas A y B de la figura 4.12.*

Solución:

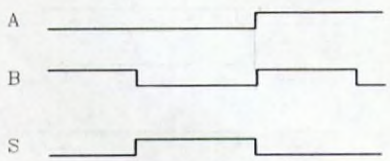


Figura 4.12

*## **Ejemplo 4.9**: Obtener el cronograma de salida de la puerta lógica NAND de dos entradas cuyos valores están representados por los cronogramas A y B de la figura 4.13.*

Solución:

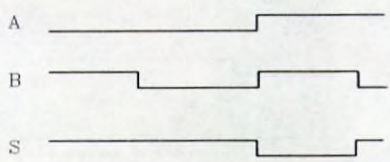


Figura 4.13

***## Ejemplo 4.10**: Obtener el cronograma de salida de la puerta lógica XOR de dos entradas cuyos valores están representados por los cronogramas A y B de la figura 4.14.*

Solución:

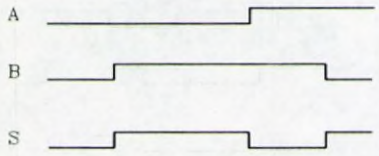


Figura 4.14

***## Ejemplo 4.11**: Obtener el cronograma de salida de la puerta lógica XNOR de dos entradas cuyos valores están representados por los cronogramas A y B de la figura 4.15.*

Solución:

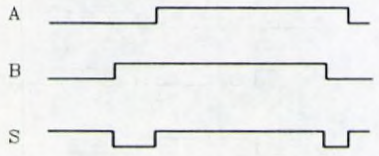


Figura 4.15

***## Ejemplo 4.12**: Obtener el cronograma de salida de la puerta lógica NO cuya entrada está representada por el cronograma A de la figura 4.16.*

Solución:

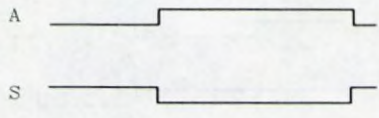


Figura 4.16

***## Ejemplo 4.13**: Obtener el cronograma de salida de la puerta lógica OR de tres entradas cuyos valores están representados por los cronogramas A, B y C de la figura 4.17.*

Solución:

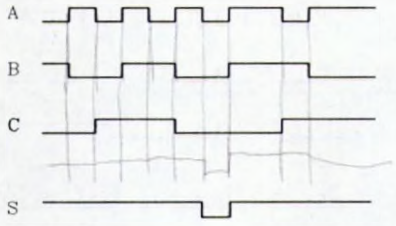


Figura 4.17

***## Ejemplo 4.14**: Obtener el cronograma de salida de la puerta AND de 3 entradas A, B y C cuyos valores se representan mediante los cronogramas de la figura 4.18.*

Solución:

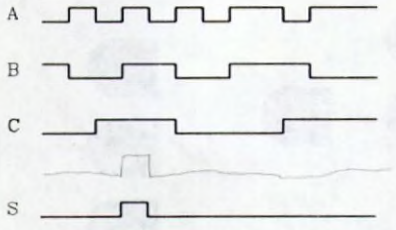


Figura 4.18

***## Ejemplo 4.15**: Obtener el cronograma de salida de la puerta NOR de 3 entradas A, B y C cuyos valores se representan mediante los cronogramas de la figura 4.19.*

Solución:

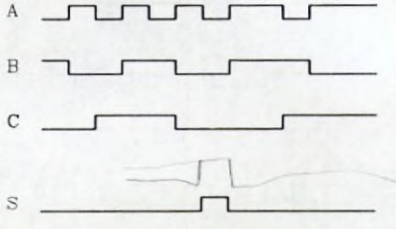


Figura 4.19

*## **Ejemplo 4.16**: Obtener el cronograma de salida de la puerta NAND de 3 entradas A, B y C cuyos valores se representan mediante los cronogramas de la figura 4.20.*

Solución:

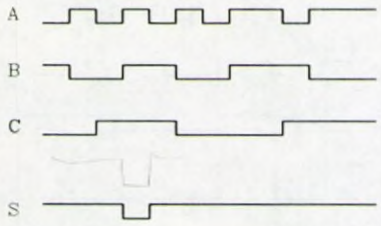


Figura 4.20

*## **Ejemplo 4.17**: Obtener el cronograma de salida del circuito de 3 entradas A, B y C de la figura 4.21 cuyos valores se representan mediante los cronogramas de la figura 4.22.*

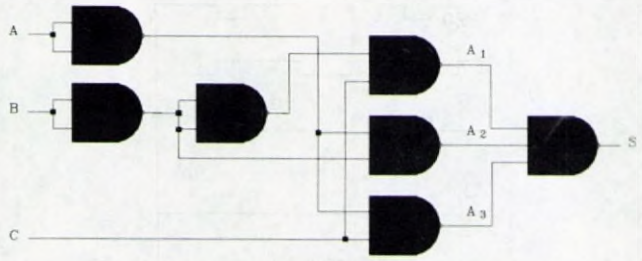


Figura 4.21

Solución:

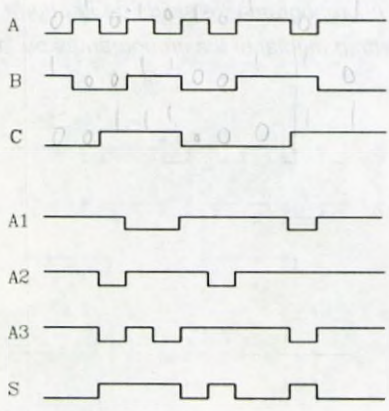


Figura 4.22

*## **Ejemplo 4.18**: Obtener el cronograma de salida del circuito de la figura 4.23 cuyas entradas se representan mediante los cronogramas de la figura 4.24.*

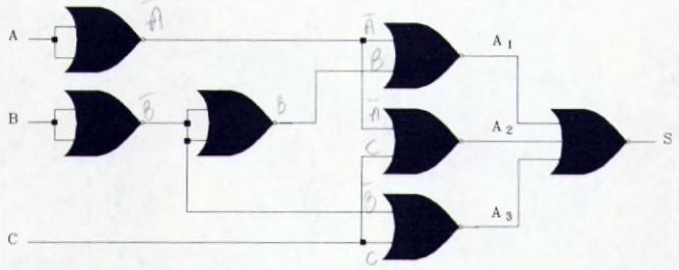


Figura 4.23

Solución:

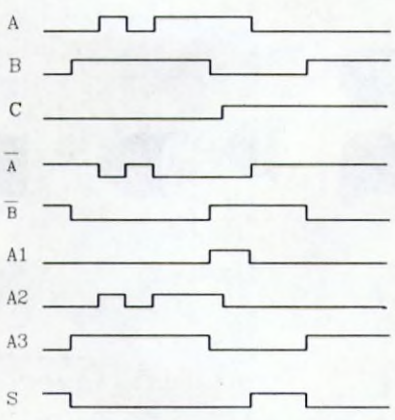


Figura 4.24

*## **Ejemplo 4.19**: Obtener la tabla de verdad de una puerta NAND de 3 entradas*

Solución:

A	B	C	SALIDA
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

*## **Ejemplo 4.20**: Obtener la tabla de verdad de una puerta NOR de 3 entradas*

Solución:

A	B	C	SALIDA
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

*## **Ejemplo 4.21**: Obtener la ecuación algebraica de salida del circuito de la figura 4.25*

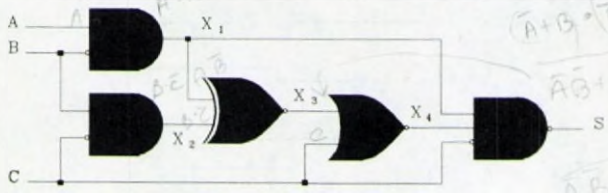


Figura 4.25

Solución:

$$X_1=A\bar{B} \qquad X_3=A\bar{B}\odot B\bar{C}=\overline{\overline{(A\bar{B})(B\bar{C})}+\overline{(A\bar{B})(\overline{B\bar{C}})}}=\bar{A}\bar{B}+\bar{A}C+BC$$

$$X_2=B\bar{C} \qquad X_4=\overline{X_3+\bar{C}}=\overline{\bar{A}\bar{B}+\bar{C}}=A\bar{C}+B\bar{C}$$

$$S=\overline{X_1X_4\bar{C}}=A\bar{B}\bar{C}$$

*## **Ejemplo 4.22**: Obtener la ecuación algebraica de salida del circuito de la figura 4.26*

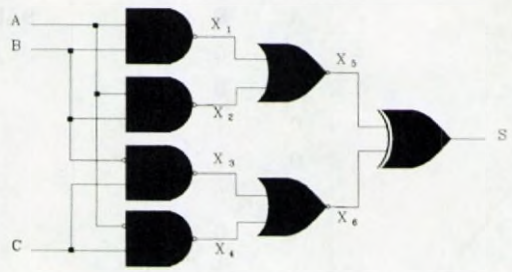


Figura 4.26

Solución:

$$X_1 = \overline{AB} \qquad X_5 = \overline{\overline{AB} + \overline{AB}} = AB$$

$$X_2 = \overline{AB} \qquad X_6 = \overline{\overline{BC} + \overline{AC}} = \overline{A}\,\overline{B}C$$

$$X_3 = \overline{\overline{BC}}$$

$$X_4 = \overline{\overline{AC}} \qquad S = (\overline{AB})(\overline{A}\,\overline{B}C) + (AB)(\overline{\overline{A}\,\overline{B}C}) = \overline{A}\,\overline{B}C + AB$$

***## Ejemplo 4.23**: Reducir el circuito de puertas lógicas de la figura 4.27*

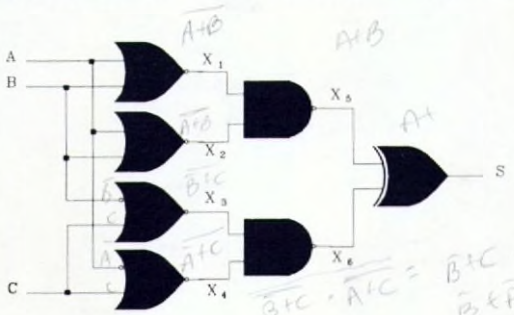


Figura 4.27

Solución:

$$X_1 = \overline{A + B} \qquad X_4 = \overline{\overline{A} + C}$$

$$X_2 = \overline{A + B} \qquad X_5 = \overline{(\overline{A} + B)(A + \overline{B})} = A + B$$

$$X_3 = \overline{\overline{B} + C} \qquad X_6 = \overline{(\overline{B} + C)(\overline{A} + C)} = \overline{A}\,\overline{B} + C$$

$$S = (\overline{A + B})(\overline{A} + \overline{B} + C) + (A + B)(\overline{\overline{A} + \overline{B} + C}) = \overline{A}\,\overline{B} + AB\overline{C}$$

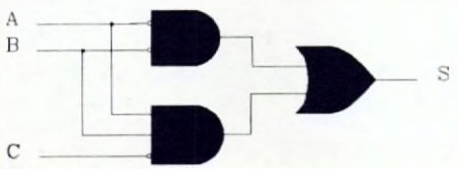


Figura 4.28

*## **Ejemplo 4.24**: Obtener la salida del circuito de la figura 4.29 cuando sus entradas toman valor "1".*

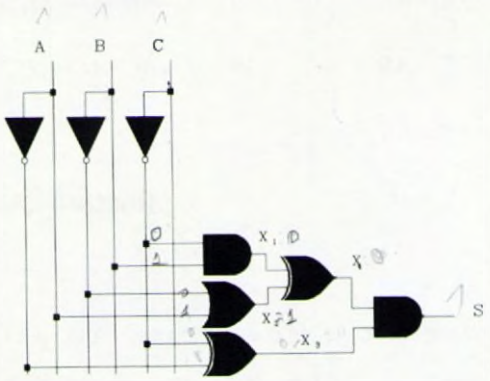


Figura 4.29

Solución:

$X_1=0 \quad X_2=1 \quad X_3=0 \quad X_4=0$

SALIDA S=1

*## **Ejemplo 4.25**: Obtener la salida del circuito de la figura 4.30 cuando sus entradas toman valor "0".*

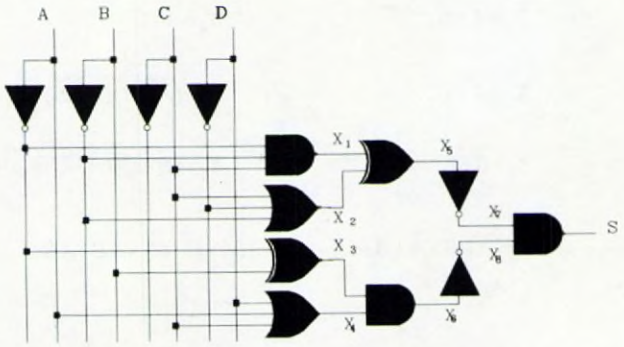


Figura 4.30

Solución:

$X_1=1 \quad X_2=0 \quad X_3=0 \quad X_4=0 \quad X_5=1 \quad X_6=0 \quad X_7=0 \quad X_8=1$

SALIDA S=1

***## Ejemplo 4.26**: Se genera un escalón de 5 ^{V} en t=2^{seg} y otro en t=4,5^{seg}. La resta de ambas señales se aplica a una puerta lógica inversora. Obtener gráficamente la salida.*

Solución:

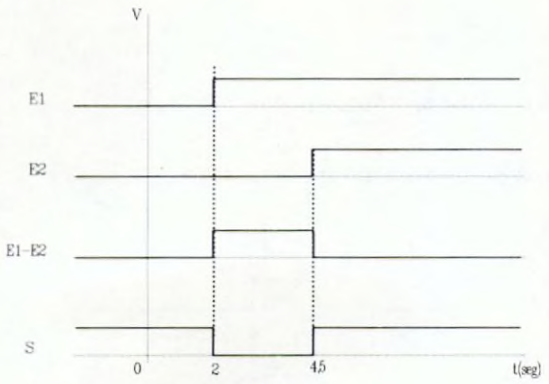


Figura 4.31

***## Ejemplo 4.27**: En t=1^{seg} se genera un escalón de 5^{V}, en t=2^{seg} se le resta otro escalón de igual tensión. En t=2,5^{seg} se le añade un tercero y en t=3,5^{seg} se resta un cuarto escalón, ambos de igual valor al primero. La señal obtenida junto con el cuarto escalón se aplican a una puerta lógica XOR. Obtener gráficamente la salida.*

Solución:

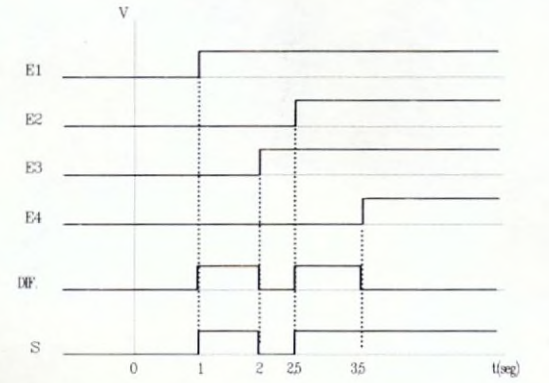


Figura 4.32

*## **Ejemplo 4.28**: Se analiza durante 15^{seg} el funcionamiento de un circuito. Si un pulso de 3^{seg} de duración se aplica a una de las entradas de una puerta OR, calcular el tiempo que dicha puerta lógica proporciona un "1" en su salida. Repetir el mismo cálculo para una puerta XNOR. Representar gráficamente todas las señales.*

Solución:

Tiempo de salida "1" en la puerta OR: 3 seg.

Tiempo de salida "1" en la puerta XNOR: 12 seg.

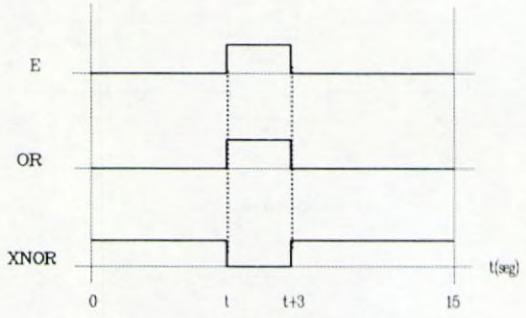


Figura 4.33

TEMA 5
SIMPLIFICACION DE FUNCIONES

5

SIMPLIFICACION DE FUNCIONES

5.1. INTRODUCCION

El tema que se analiza a continuación se puede considerar como una segunda parte del tema anterior de álgebra booleana, ampliando conocimientos y elevando el nivel de complejidad de los circuitos y elementos. En este tema, se constituyen las funciones algebraicas y se relacionan con los valores numéricos, introduciendo las dos variantes de expresión y análisis denominadas formas canónicas.

El análisis de funciones, completa o incompletamente especificadas, conducirá siempre a la representación y uso de las tablas de verdad. Las dos formas canónicas expresan de forma más simple una función lógica, empleándose fundamentalmente como herramienta rápida de operación. Al mismo tiempo, sirven de introducción a la simplificación de funciones lógicas. Se analizan los procedimientos de simplificación o minimización de Karnaugh y Quine McCluskey, describiendo las distintas operaciones, construyendo las tablas correspondientes y terminando con la construcción del circuito de puertas lógicas, tanto empleando cualquier tipo de puerta como generalizando en un único modelo.

5.2. FUNCIONES Y TABLAS DE VERDAD

En el tema anterior se aplicó el concepto de función a las operaciones de suma y producto. A continuación se describen las funciones como concepto genérico, utilizándose y aplicándose a cualquier diseño lógico y no sólo a operaciones aritméticas. La función se constituye genéricamente mediante las combinaciones de una serie de variables, considerando que estas variables se corresponden con las entradas de datos lógicos mientras que la función se corresponde con la salida. Por tanto, la expresión de dicha función equivaldrá a una formulación algebraica, donde los valores de las variables de entrada

estarán indicados mediante la propia variable o su complemento, dependiendo que se intente expresar unos valores lógicos correspondientes al "1" ó al "0".

Si la expresión A complementada equivale al "0" y A al "1", cada una de las variables se equipara a un bit del sistema binario, pudiendo existir tantas combinaciones de variables como las ya conocidas de los dígitos del sistema binario. Si para dicho sistema de numeración se expresan las combinaciones en forma de tabla de valores, esta tabla aplicada a las variables lógicas y relacionadas con una función se denominará tabla de verdad.

Los términos algebraicos se pueden expresar mediante términos numéricos, pero eso no significa que puedan operar aritméticamente, sino que tan sólo corresponde a una representación lógica. A cada uno de los términos que definen una función F (empleando expresiones de suma de términos), se le asigna un "1" en la construcción de la tabla de verdad, mientras que los términos que no aparecen en dicha expresión llevan asignado un "0". De esta forma se obtiene la tabla que relaciona las variables de entrada y la función F.

*## **Ejemplo 5.1**: Representar en forma numérica y mediante tabla de verdad la función de 4 variables cuya expresión algebraica es la siguiente:*

$$F(A,B,C,D)=\bar{A}BCD+A\bar{B}CD+\bar{A}\bar{B}CD+\bar{A}B\bar{C}D+A\bar{B}\bar{C}\bar{D}$$

Solución:

Expresión numérica:

$$F(A,B,C,D)=1111+1011+0111+0101+1010$$

Tabla de verdad:

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

*## **Ejemplo 5.2**: Simplificar mediante álgebra de Boole la función del problema 5.1.*

Solución:

$$F=ABCD+A\overline{B}CD+\overline{A}BCD+\overline{A}\overline{B}CD+A\overline{B}\overline{C}\overline{D}$$

$$F=ACD+\overline{A}BD+A\overline{B}C\overline{D}=ACD+\overline{A}BD+A\overline{B}C$$

5.3. EXPRESION MEDIANTE FORMAS CANONICAS

Se denominan formas canónicas de **SUMA DE PRODUCTOS** y **PRODUCTO DE SUMAS** a unas expresiones simbólicas equivalentes a las representaciones algebraicas de una función, cuyos términos estarán simbolizados con las letras **"m"** y **"M"**.

Suma de productos es la expresión empleada para la **PRIMERA FORMA CANONICA**, y se corresponde con el desarrollo utilizado en los ejemplos 5.1 y 5.2, donde las variables algebraicas se multiplican lógicamente entre sí, agrupándose en términos que se relacionan mediante la operación de suma lógica. Sus términos se simplifican y simbolizan mediante la letra "m", añadiéndole el subíndice correspondiente al valor decimal de la combinación binaria que representan. Cada término "m" corresponde a un "1" de la columna F de la tabla de verdad. Quiere decir por tanto, que una función F cualquiera se puede representar mediante la tabla de verdad, la expresión algebraica, la expresión numérica o mediante expresión en términos canónicos.

La **SEGUNDA FORMA CANONICA** se denomina producto de sumas. Los términos se simbolizan mediante la letra "M", y el proceso de obtención de los mismos es el siguiente:

* Se relacionan los elementos que hacen que la función F tome el valor "0"

* Se calcula su complemento a $\mathbf{2^z-1}$ (valor máximo de la tabla), siendo "z" el número de variables que intervienen en la definición de la función F. (El complemento es la diferencia entre 2^z-1 y el número correspondiente)

* Los complementos obtenidos según la regla anterior serán los subíndices de los coeficientes "M".

Una vez obtenidos los términos de la segunda forma canónica, la función F vendrá representada por el producto de los mismos. La expresión de una función mediante producto de sumas puede realizarse también en las formas

numérica y algebraica, tal como se realiza para la suma de productos, teniendo en cuenta que las operaciones entre las variables (suma y producto lógico) estarán complementadas.

*## **Ejemplo 5.3**: Representar mediante suma de productos la función del ejemplo 5.1.*

Solución:

$$F(A,B,C,D)=m_5+m_7+m_{10}+m_{11}+m_{15}$$

*## **Ejemplo 5.4**: Representar mediante producto de sumas la función del ejemplo 5.1.*

Solución:

$$F(A,B,C,D)=M_1M_2M_3M_6M_7M_9M_{11}M_{12}M_{13}M_{14}M_{15}$$

*## **Ejemplo 5.5**: Representar mediante tabla de verdad, producto de sumas, suma de productos, expresiones algebraicas y numéricas la siguiente función:*

$$F(A,B,C,D)=A\overline{B}CD+AB\overline{C}D+\overline{A}BCD+\overline{A}BC\overline{D}+A\overline{B}\overline{C}\overline{D}+ABC\overline{D}+ABCD+\overline{A}\overline{B}\overline{C}D$$

Solución:

Tabla de verdad:

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Expresión canónica de suma de productos:

$$F(A,B,C,D)=m_5+m_6+m_7+m_{10}+m_{11}+m_{13}+m_{14}+m_{15}$$

Expresión numérica de suma de productos:

$$F(A,B,C,D)=0101+1011+1101+0111+0110+1010+1110+1111$$

Expresión canónica de producto de sumas:

$$F(A,B,C,D)=M_3M_6M_7M_{11}M_{12}M_{13}M_{14}M_{15}$$

Expresión numérica de producto de sumas:

$$F=(0+0+1+1)(0+1+1+0)(0+1+1+1)(1+0+1+1)(1+1+0+0)(1+1+0+1)(1+1+1+0)(1+1+1+1)$$

Expresión algebraica de producto de sumas:

$$F=(\overline{A}+\overline{B}+C+D)(\overline{A}+B+C+\overline{D})(\overline{A}+B+C+D)(A+\overline{B}+C+D)(A+B+\overline{C}+\overline{D})(A+B+\overline{C}+D)(A+B+C+\overline{D})(A+B+C+D)$$

*## **Ejemplo 5.6**: Representar mediante tabla de verdad, producto de sumas y suma de productos la siguiente función:*

$$F(A,B,C,D)=A\overline{B}\overline{C}\overline{D}+\overline{A}B\overline{C}\overline{D}+AB\overline{C}D+\overline{A}BC\overline{D}+ABC\overline{D}+ABCD+A\overline{B}CD$$

Solución:

Tabla de verdad:

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Expresión canónica de suma de productos:

$$F(A,B,C,D)=m_5+m_6+m_{10}+m_{11}+m_{13}+m_{14}+m_{15}$$

Expresión canónica de producto de sumas:

$$F(A,B,C,D)=M_3M_6M_7M_8M_{11}M_{12}M_{13}M_{14}M_{15}$$

*## **Ejemplo 5.7**: Representar mediante tabla de verdad, producto de sumas y suma de productos la siguiente función de 4 variables:*

$$F=(A+\overline{B}+C+\overline{D})(\overline{A}+B+\overline{C}+D)(A+B+\overline{C}+D)(\overline{A}+B+C+\overline{D})(A+B+C+\overline{D})(A+B+C+D)$$

Solución:

Tabla de verdad:

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Expresión canónica de suma de productos:

$$F(A,B,C,D)=m_3+m_4+m_6+m_7+m_8+m_{11}+m_{12}+m_{13}+m_{14}+m_{15}$$

Expresión canónica de producto de sumas:

$$F(A,B,C,D)=M_5M_6M_{10}M_{13}M_{14}M_{15}$$

5.4. SIMPLIFICACION POR EL METODO DE KARNAUGH

A medida que aumenta el número de variables de entrada de una función, la simplificación de los términos que la definen es cada vez más complicada y difícil por los métodos tradicionales del álgebra booleana. Se necesitan herramientas de simplificación más precisas y sencillas, por lo que se determinan para estos fines los procedimientos de simplificación o minimización de Karnaugh y Quine McCluskey.

El método de Karnaugh se basa en la construcción de unas tablas de simplificación equivalentes a las tablas de verdad aunque estructuradas en forma gráfica, en las que intervendrán las variables de entrada y las formas canónicas que representan la función que se está analizando. Dichas tablas tienen una construcción en forma progresiva, según que la función esté definida por 2, 3, 4, 5 ó más variables de entrada.

En las tablas, se dispondrán dos líneas de asignación de variables, una horizontal y otra vertical, en las que se indicarán las distintas combinaciones que pueden tomar los valores numéricos binarios que representan dichas variables, de acuerdo con la relación entre valor numérico y expresión algebraica que se ha detallado con anterioridad. Se forma una estructura de celdas elementales donde se colocarán los valores de la función correspondientes a cada combinación binaria. El orden de colocación en las tablas se puede apreciar en los gráficos de la figura 5.1., donde se han representado las tablas correspondientes a 2, 3 y 4 variables de entrada. Las combinaciones binarias entre columnas o filas adyacentes varían únicamente en un bit.

La combinación de los valores binarios de las filas horizontales y verticales, determina la totalidad de las posibles combinaciones binarias que se pueden efectuar con dos, tres, cuatro, etc. Estas combinaciones binarias, conducen a la obtención de los subíndices empleados en las dos formas canónicas. Así, y para una mayor facilidad de simplificación de expresiones, cada celda estará representada por una combinación binaria diferente, o lo que es lo mismo, por un número decimal diferente, que servirá de referencia para su posterior localización y asignación de valores canónicos. En la figura 5.1 se han representado las tablas de Karnaugh asignándole a cada celda el valor decimal correspondiente a la combinación binaria que la determina, disponiendo de las siguientes posibilidades para la asignación de valores de la función:

* Si se emplea la primera forma canónica de suma de productos, se asignará a cada casilla el valor correspondiente de la función en la tabla de verdad. Si cada línea de la tabla de verdad de la función corresponde a un término "m", el subíndice de cada uno de dichos términos corresponde a una casilla de la tabla de Karnaugh. Se colocará por tanto en cada casilla

de la tabla de Karnaugh el número binario "0" ó "1" que corresponda a la columna F de la tabla de verdad.

* Si se emplea la segunda forma canónica de producto de sumas, se asignará a cada casilla un número binario que determine el valor de la función F, similar a lo realizado para la primera forma canónica. Sin embargo, en este caso, se tienen que transformar los valores tal como se explicó para la determinación de los términos "M" de la segunda forma canónica. Si para la primera forma canónica los términos "1" de la tabla de Karnaugh se correspondían con los términos "m", en este caso de la segunda forma canónica, igualmente los términos "1" de la tabla de Karnaugh se corresponderán con los términos "M", aunque éstos no se corresponden ahora con los valores "1" de la tabla de verdad. Es sólo una facilidad operativa. Los restantes casilleros se rellenarán con el valor "0". (Es práctica usual no escribir los ceros para mayor claridad ya que no son necesarios en la operatividad final)

B \ A	0	1
0	0	2
1	1	3

C \ A B	00	01	11	10
0	0	2	6	4
1	1	3	7	5

C D \ A B	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

Figura 5.1. Tablas de Karnaugh

El procedimiento de simplificación consiste en **AGRUPAR LAS CELDAS CONTIGUAS CON TERMINOS "1"** ó que **VARIAN ENTRE SI UNICAMENTE EN UNO DE LOS BITS O VARIABLES** que las definen (**FILAS Y COLUMNAS EXTREMAS**), de tal manera que se puedan ir eliminando variables. **SOLO SE PUEDEN AGRUPAR UN NUMERO DE CELDAS IGUAL A UNA POTENCIA DE BASE 2** (1, 2, 4, 8, 16, etc.). Un término puede pertenecer a varias agrupaciones. Se pueden agrupar todos los términos de la tabla dando como resultado el valor "1" para la función.

En las figuras 5.2 se esquematizan las diferentes agrupaciones de elementos que se pueden realizar en las tablas de Karnaugh. La agrupación de dos elementos de la tabla simplifica una variable, la de cuatro términos simplifica dos variables, la de ocho términos simplifica tres variables, y así suce-

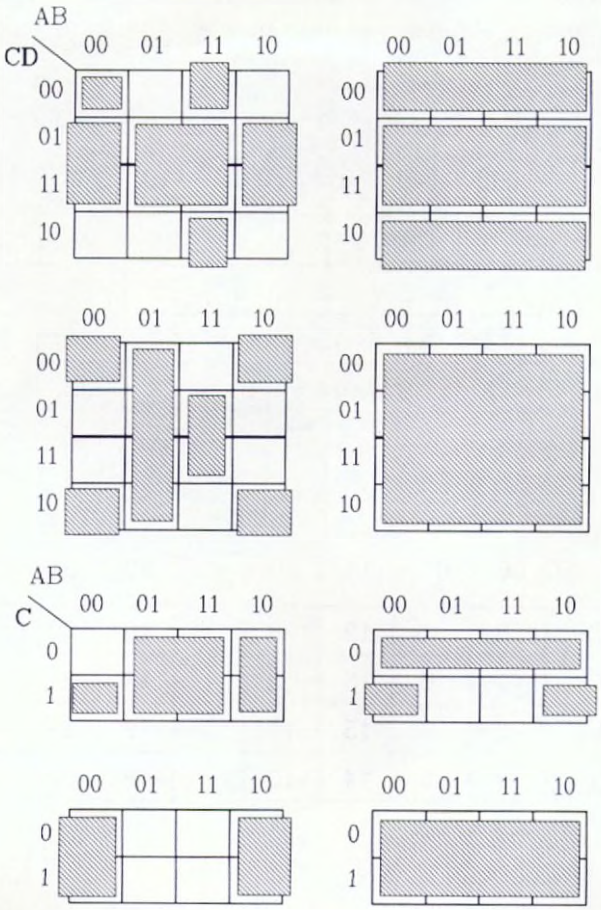


Figura 5.2. Agrupaciones en términos de Karnaugh

sivamente a medida que aumenta el número de términos simplificados. El objeto del método es buscar la solución mínima algebraica que significará el diseño del circuito más sencillo posible. Las expresiones algebraicas que se obtienen a partir de las tablas de Karnaugh están formadas únicamente por las variables de fila y columna que se repiten en todos los términos de la agrupación. Las variables que toman ambos valores ("0" y "1") se simplifican de la expresión.

Cuando se analizan los diagramas de simplificación de Karnaugh, normalmente se circunscriben a un máximo de 4 variables, dado que a partir de ahí se complican las tablas y el proceso de obtención de la expresión algebraica simplificada. El proceso de reducción de funciones de 5 y 6 variables, valores frecuentes en el funcionamiento de los circuitos, y al mismo tiempo, número límite de variables para tener que pasar forzosamente al análisis por el método tabular y sistemático de Quine McCluskey, es similar al proceso genérico descrito, pero existen algunas variantes que facilitan su resolución.

D E \ A B C	000	001	011	010	110	111	101	100
00	0	4	12	8	24	28	20	16
01	1	5	13	9	25	29	21	17
11	3	7	15	11	27	31	23	19
10	2	6	14	10	26	30	22	18

D E \ B C	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

$\overline{A}$

00	01	11	10
16	20	28	24
17	21	29	25
19	23	31	27
18	22	30	26

A

Figura 5.3. Tablas de Karnaugh de 5 variables

Para el caso de 5 variables, se forma una tabla de 32 valores según se puede observar en la figura 5.3, numeradas según el equivalente binario de las variables. Esta tabla genérica se puede desdoblar en otras dos más pequeñas y manejables. La tabla de la derecha corresponde a la variable A y la de la izquierda a la misma variable complementada, teniendo ambas las mismas combinaciones binarias (igual orden de los términos de las variables B y C). Se ha separado la variable A por comodidad, pudiendo separarse cualquier otra variable de las que definen la función.

Analizando ambas tablas simultáneamente, se puede comprobar que si una misma casilla está ocupada por un "1" en ambas tablas, se elimina la variable A en la expresión algebraica de dicha función. Si no se repiten los términos en ambas tablas, cada uno de ellos vendrá expresado por las variables que le correspondan además de la A o su complemento según esté dicho término a la derecha o a la izquierda.

Para el caso de 6 variables, se forma una tabla de 64 valores según se puede observar en la figura 5.4, numeradas según el equivalente binario de las variables. Esta tabla genérica se puede desdoblar en otras cuatro más pequeñas. Las tablas de la derecha corresponden a la variable A y las de la izquierda a la misma variable complementada, teniendo ambas las mismas combinaciones binarias (igual orden de los términos de las variables B y C) mientras que las tablas de la fila superior corresponden a la variable D y las de la inferior a la misma variable complementada. De esta forma, cada una de las cuatro tablas está representada por dos variables externas, además de las cuatro que constituyen los valores de las mismas. Igual que para las tablas de 5 variables, se han separado las variables A y D por comodidad, pudiendo separarse cualquier otra variable de las que definen la función. Analizando las tablas simultáneamente, se puede comprobar que si una misma casilla está ocupada por un "1" en todas las tablas, se eliminan las variables A y D en la expresión algebraica de dicha función. Si la ocupación es equivalente en dos de las cuatro tablas, se puede eliminar una variable siempre que se correspondan en la misma fila o columna. Si no se repiten los términos en ninguna de las tablas, cada uno de ellos vendrá expresado por las variables que le correspondan además de la A y D o sus complementos según esté dicho término colocado.

D E F \ A B C	000	001	011	010	110	111	101	100
000	0	8	24	16	48	56	40	32
001	1	9	25	17	49	57	41	33
011	3	11	27	19	51	59	43	35
010	2	10	26	18	50	58	42	34
110	6	14	30	22	54	62	46	38
111	7	15	31	23	55	63	47	39
101	5	13	29	21	53	61	45	37
100	4	12	28	20	52	60	44	36

E F \ B C	00	01	11	10
00	0	8	24	16
01	1	9	25	17
11	3	11	27	19
10	2	10	26	18

$\overline{A}$

E F \ B C	00	01	11	10
00	32	40	56	48
01	33	41	57	49
11	35	43	59	51
10	34	42	58	50

$\overline{D}$

E F \ B C	00	01	11	10
00	4	12	28	20
01	5	13	29	21
11	7	15	31	23
10	6	14	30	22

$\overline{A}$

E F \ B C	00	01	11	10
00	36	44	60	52
01	37	45	61	53
11	39	47	63	55
10	38	46	62	54

D

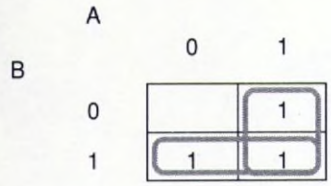
A

Figura 5.4. Tablas de Karnaugh de 6 variables

*## **Ejemplo 5.8**: Simplificar por el método de Karnaugh empleando suma de productos la siguiente función algebraica de 2 variables:*

$$F(A,B)=\overline{A}B+A\overline{B}+AB$$

Solución:

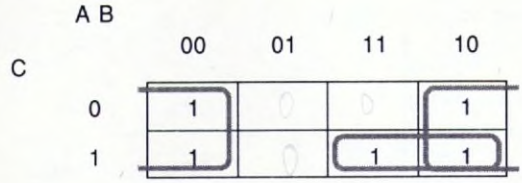


$$F(A,B)=A+B$$

*## **Ejemplo 5.9**: Simplificar por el método de Karnaugh empleando suma de productos la siguiente función algebraica de 3 variables:*

$$F(A,B,C)=ABC+A\overline{B}C+A\overline{B}\overline{C}+\overline{A}\overline{B}C+\overline{A}\overline{B}\overline{C}$$

Solución:

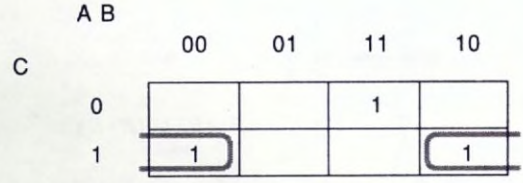


$$F(A,B,C)=\overline{B}+AC$$

*## **Ejemplo 5.10**: Simplificar por el método de Karnaugh empleando producto de sumas la siguiente función algebraica de 3 variables:*

$$F(A,B,C,)=\overline{A}\overline{B}\overline{C}+\overline{A}BC+A\overline{B}\overline{C}+A\overline{B}C+ABC$$

Solución: $F=M_1M_5M_6$



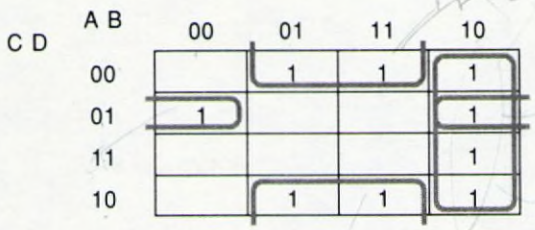
$$F(A,B,C)=(\overline{B}+C)(A+B+\overline{C})$$

*## **Ejemplo 5.11**: Simplificar por el método de Karnaugh empleando producto de sumas la siguiente función algebraica de 4 variables:*

$$F=m_0+m_2+m_8+m_{10}+m_{12}+m_{13}+m_{15}$$

Solución:

$$F=M_1M_4M_6M_8M_9M_{10}M_{11}M_{12}M_{14}$$

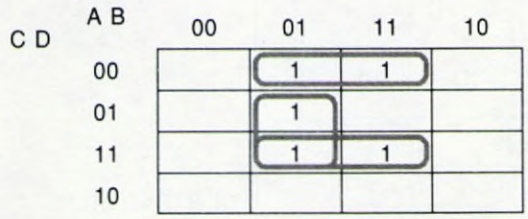


$$F=(A+\overline{B})(B+\overline{D})(\overline{B}+\overline{C}+D)$$

*## **Ejemplo 5.12**: Simplificar por el método de Karnaugh empleando suma de productos la siguiente función algebraica de 4 variables:*

$$F=\overline{A}B\overline{C}\overline{D}+\overline{A}B\overline{C}D+\overline{A}BCD+AB\overline{C}\overline{D}+ABCD$$

Solución:



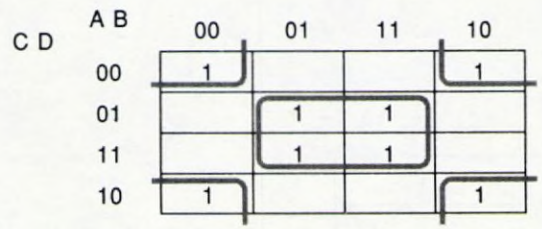
$$F=B\overline{C}\overline{D}+BCD+\overline{A}BD$$

*## **Ejemplo 5.13**: Simplificar por el método de Karnaugh empleando producto de sumas la siguiente función algebraica de 4 variables:*

$$F=m_1+m_3+m_4+m_6+m_9+m_{11}+m_{12}+m_{14}$$

Solución:

$$F=M_0M_2M_5M_7M_8M_{10}M_{13}M_{15}$$

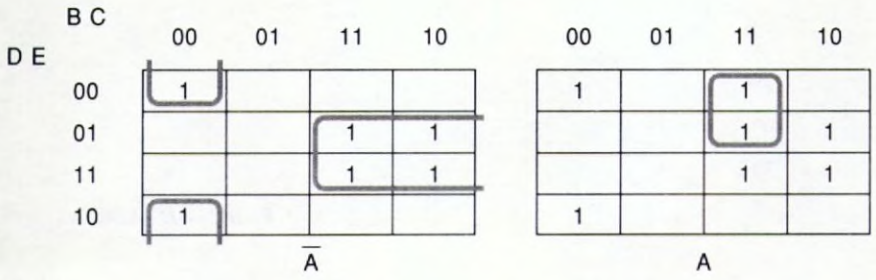


$$F=(B+D)(\overline{B}+\overline{D})$$

*## **Ejemplo 5.14**: Simplificar por el método de Karnaugh empleando suma de productos la siguiente función de 5 variables:*

$$F=m_0+m_2+m_9+m_{11}+m_{13}+m_{15}+m_{16}+m_{18}+m_{25}+m_{27}+m_{28}+m_{29}+m_{31}$$

Solución:

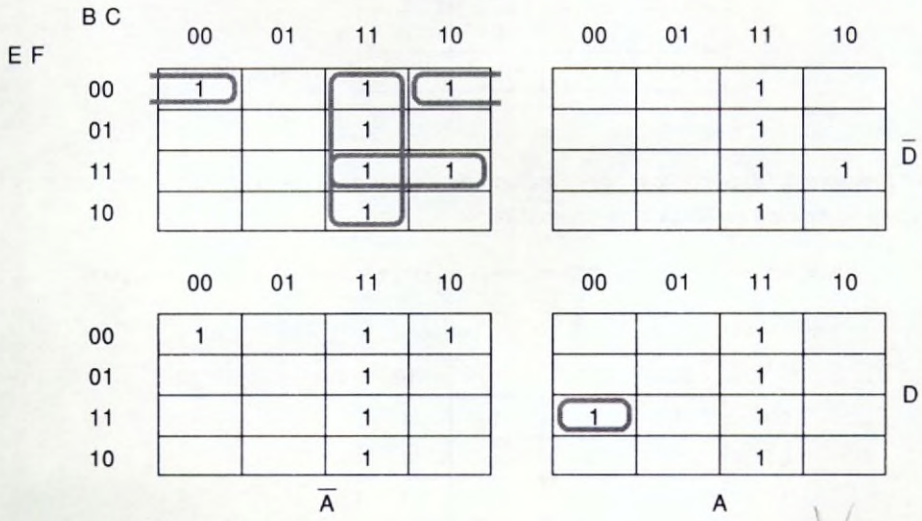


$$F=\bar{B}\bar{C}\bar{E}+BE+ABC\bar{D}$$

*## **Ejemplo 5.15**: Simplificar por el método de Karnaugh empleando suma de productos la siguiente función de 6 variables:*

$$F=m_0+m_4+m_{16}+m_{19}+m_{20}+m_{24}+m_{25}+m_{26}+m_{27}+m_{28}+m_{29}+$$
$$m_{30}+m_{31}+m_{39}+m_{51}+m_{56}+m_{57}+m_{58}+m_{59}+m_{60}+m_{61}+m_{62}+m_{63}$$

Solución:



$$F=BC+\bar{A}\bar{C}\bar{E}\bar{F}+B\bar{D}EF+A\bar{B}\bar{C}DEF$$

*## **Ejemplo 5.16**: Simplificar por el método de Karnaugh empleando suma de productos la siguiente función algebraica de 4 variables:*

$$F=M_0M_1M_2M_3M_7M_8M_9M_{10}M_{15}$$

Solución: $F=m_1+m_2+m_3+m_4+m_9+m_{10}+m_{11}$

C D \ A B	00	01	11	10
00		1		
01	1			1
11	1			1
10	1			1

$$F=\bar{B}C+\bar{B}D+\bar{A}B\bar{C}\bar{D}$$

*## **Ejemplo 5.17**: Simplificar por el método de Karnaugh empleando suma de productos la siguiente función algebraica de 4 variables:*

$$F=M_0M_1M_2M_3M_4M_5M_6M_7M_8M_9M_{10}M_{11}M_{12}M_{13}M_{14}M_{15}$$

Solución: No existen términos en suma de productos
No hay términos "1" en la tabla de Karnaugh

C D \ A B	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	0	0
10	0	0	0	0

$$F(A,B,C,D)=0$$

*## **Ejemplo 5.18**: Simplificar por el método de Karnaugh empleando suma de productos la siguiente función algebraica de 4 variables:*

$$F=m_0+m_1+m_2+m_3+m_4+m_5+m_6+m_7+m_8+m_9+m_{10}+m_{11}+m_{12}+m_{13}+m_{14}+m_{15}$$

Solución:

C D \ A B	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

$$F(A,B,C,D)=1$$

5.5. SIMPLIFICACION POR EL METODO DE QUINE McCLUSKEY

El segundo método de minimización de funciones lógicas se diferencia del método de Karnaugh en que el proceso de construcción de las tablas que lo definen, es idéntico para todo número de variables de entrada. Si las tablas de Karnaugh eran diferentes según las variables que intervinieran, las tablas de Quine McCluskey son siempre las mismas. Este segundo método para un número reducido de variables (menos de cinco), es bastante más lento que el anterior, aunque a medida que aumenta el número de variables, su empleo resulta más ventajoso. Es un método lento, pero sistemático y seguro, realizándose una serie ordenada de etapas de simplificación, que redundan en una clara visión de los elementos que se van agrupando en la minimización. Al igual que para el método de Karnaugh, en este método se emplean las dos expresiones o formas canónicas definidas anteriormente, las de suma de productos y producto de sumas.

Partiendo de la tabla de verdad que define la función a estudiar se extraen las combinaciones binarias que se corresponden con un "1" de la función F, o lo que es lo mismo, se extraerán los términos "m" correspondientes a la primera forma canónica. Estas combinaciones binarias se ordenarán según el número de elementos "1" que contengan, formando una serie de grupos de orden cero, uno, dos, etc., según el número de variables que intervienen. Los bloques de orden consecutivo se irán simplificando de dos en dos, en forma contigua, es decir, el de orden cero con el de orden uno, el de orden uno con el de orden dos, etc., formando nuevas agrupaciones.

La simplificación se irá realizando entre combinaciones binarias que tengan tres elementos coincidentes, definiéndose la nueva combinación resultante de la simplificación mediante los tres elementos coincidentes y un guión que ocupa la posición del cuarto elemento (único que debe variar entre ambas combinaciones). El guión se corresponde con la simplificación de una de las variables que intervienen en la definición de la función correspondiente. Una vez formado un nuevo grupo de tablas, se procederá a repetir el proceso de simplificación, buscando combinaciones entre grupos de orden consecutivo, con tres elementos coincidentes, incluyendo los guiones procedentes de la primera simplificación. El tercer grupo de tablas que se forma, contendrá dos guiones en cada una de sus combinaciones, correspondiente a la simplificación de dos variables de entrada. El proceso se seguirá repitiendo mientras haya combinaciones que puedan ser simplificadas. Una vez terminado el proceso de simplificación de combinaciones binarias, será necesario comparar el último conjunto de agrupaciones obtenido, con el primero que se formó con las expresiones que definían un "1" en la función F.

Para ello se crea una nueva tabla donde se sitúan las combinaciones iniciales que determinaban un "1" en F. En la línea horizontal se sitúan los términos que definen F, y las últimas combinaciones simplificadas obtenidas en las tablas de Quine McCluskey se sitúan en la línea vertical. Se marcarán en la tabla (por ejemplo con una "x"), las distintas combinaciones originales que queden abarcadas por las últimas combinaciones simplificadas obtenidas, teniendo en cuenta únicamente los elementos numéricos que aún queden en dichas combinaciones. Si no se abarcaran todos los términos originales, será necesario recurrir a la tabla anterior para tomar las combinaciones simplificadas necesarias que permitan disponer de todas las combinaciones iniciales. En caso de utilizar la segunda forma canónica, la de producto de sumas, el procedimiento a realizar sería el mismo, pero se emplearían las combinaciones binarias definidas por los términos "M". El resultado final estará expresado como producto de sumas. Antes de expresar la función algebraica final es necesario comprobar si algún término puede ser eliminado sin que por ello se dejen de cubrir todos los elementos originales (Ver ejemplo 5.39.S4).

*## **Ejemplo 5.19**: Simplificar por el método de Quine McCluskey utilizando suma de productos la siguiente función algebraica de 4 variables:*

$$F = m_0 + m_1 + m_3 + m_4 + m_7 + m_9 + m_{11} + m_{12} + m_{14} + m_{15}$$

Solución:

0000	000-	-0-1
------	0-00	------
0001	------	--11
0100	00-1	
------	-001	
0011	-100	
1001	------	
1100	0-11	
------	-011	
0111	10-1	
1011	11-0	
1110	------	
------	-111	
1111	1-11	
	111-	

	0000	0001	0100	0011	1001	1100	0111	1011	1110	1111
-0-1		x		x	x			x		
--11				x			x	x		x
0-00	x		x							
11-0						x			x	

$$F=\bar{B}D+CD+\bar{A}\bar{C}\bar{D}+AB\bar{D}$$

***## Ejemplo 5.20**: Simplificar por el método de Quine McCluskey utilizando producto de sumas la siguiente función de 4 variables:*

$$F=m_0+m_3+m_5+m_8+m_{10}+m_{13}+m_{15}$$

Solución:

0001	00-1	-0-1
0100	-001	
1000	01-0	
------	100-	
0011	------	
0110	-011	
1001	-110	
------	10-1	
1011	1-01	
1101		
1110		

	0001	0100	1000	0011	0110	1001	1011	1101	1110
-0-1	x			x		x	x		
01-0		x			x				
100-			x			x			
1-01						x		x	
-110					x				x

$$F=(\bar{B}+D)(\bar{A}+B+\bar{D})(A+\bar{B}+\bar{C})(A+\bar{C}+D)(B+C+\bar{D})$$

***## Ejemplo 5.21**: Simplificar por el método de Quine McCluskey utilizando suma de productos la siguiente función de 5 variables:*

$$F=m_0+m_1+m_3+m_6+m_8+m_9+m_{11}+m_{15}+m_{19}+m_{20}+m_{23}+m_{25}+m_{26}+m_{27}+m_{30}$$

Solución:

00000	m_0	0000-	m_0/m_1	0-00-	$m_0/m_1/m_8/m_9$
-------		0-000	m_0/m_8	-------	
00001	m_1	-------		0-0-1	$m_1/m_3/m_9/m_{11}$
01000	m_8	000-1	m_1/m_3	-------	
-------		0-001	m_1/m_9	—011	$m_3/m_{11}/m_{19}/m_{27}$
00011	m_3	0100-	m_8/m_9	-10-1	$m_9/m_{11}/m_{25}/m_{27}$
00110	m_6	-------			
01001	m_9	0-011	m_3/m_{11}		
10100	m_{20}	-0011	m_3/m_{19}		
-------		010-1	m_9/m_{11}		
01011	m_{11}	-1001	m_9/m_{25}		
10011	m_{19}	-------			
11001	m_{25}	01-11	m_{11}/m_{15}		
11010	m_{26}	-1011	m_{11}/m_{27}		
-------		10-11	m_{19}/m_{23}		
01111	m_{15}	1-011	m_{19}/m_{27}		
10111	m_{23}	110-1	m_{25}/m_{27}		
11011	m_{27}	1101-	m_{26}/m_{27}		
11110	m_{30}	11-10	m_{26}/m_{30}		

	0-00-	**0-0-1**	**--011**	**-10-1**	**11-10**	**01-11**	**10-11**	**10100**	**00110**
m_0	x								
m_1	x	x							
m_3		x	x						
m_6									x
m_8	x								
m_9	x	x		x					
m_{11}		x	x	x		x			
m_{15}						x			
m_{19}			x				x		
m_{20}								x	
m_{23}							x		
m_{25}				x					
m_{26}					x				
m_{27}			x	x					
m_{30}					x				

$$F=\bar{A}\bar{C}\bar{D}+\bar{A}\bar{C}E+\bar{C}DE+B\bar{C}E+ABD\bar{E}+\bar{A}BDE+A\bar{B}DE+A\bar{B}C\bar{D}\bar{E}+\bar{A}\bar{B}CD\bar{E}$$

*## **Ejemplo 5.22**: Simplificar por el método de Quine McCluskey utilizando suma de productos la siguiente función algebraica de 3 variables:*

$$F=M_2M_3M_6M_7$$

Solución:

		010		01-		-1-	
		-----		-10			
		011		-----			
		110		-11			
		-----		11-			
		111					
		010	**011**	**110**	**111**		
	-1-	x	x	x	x		$F(A,B,C)=B$

*## **Ejemplo 5.23**: Simplificar por el método de Quine McCluskey utilizando producto de sumas la siguiente función de 4 variables:*

$$F=M_0M_3M_5M_8M_{10}M_{13}M_{15}$$

Solución:

		0000		-000				
		------		------				
		1000		10-0				
		------		------				
		0011		-101				
		0101		------				
		1010		11-1				

		1101						

		1111						
		0000	**1000**	**0011**	**0101**	**1010**	**1101**	**1111**
	-000	x	x					
	10-0		x			x		
	-101				x		x	
	11-1						x	x
	0011			x				

$$F=(\overline{B}+\overline{C}+\overline{D})(A+\overline{B}+\overline{D})(B+\overline{C}+D)(A+B+D)(\overline{A}+\overline{B}+C+D)$$

5.6. SIMPLIFICACION DE FUNCIONES INCOMPLETAS

Se denominan funciones incompletamente especificadas o incompletas a aquellas que no están definidas para todas las combinaciones de las variables que las determinan. Si por ejemplo, tenemos una función constituida por tres variables, que pueden dar lugar a ocho posibles combinaciones diferentes, cualquier definición de la misma con un número de términos menor de ocho la delimita como incompleta. (Es preciso no confundirlo con el número de términos canónicos que aparezcan en su expresión algebraica).

Los términos no especificados se simbolizan con un guión "-" o una "x", expresión que equivale a decir que puede tomar indistintamente los valores "0" ó "1" según se requiera. Los valores que se asignan a dichas situaciones inespecificadas no alteran el funcionamiento del circuito, ya que corresponden a salidas con valores no definidos de las entradas o a entradas no relacionadas con la salida, manteniéndose por tanto inalterable el conjunto y posibilitando la obtención de una expresión y un circuito real más simple. Cualquiera de los métodos de Karnaugh y Quine McCluskey en cualquiera de sus formas canónicas pueden ser empleados para simplificar funciones con términos no especificados.

*## **Ejemplo 5.24**: Simplificar por Karnaugh la siguiente función (expresada mediante la tabla de verdad):*

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	-
1	1	1	-

Solución:

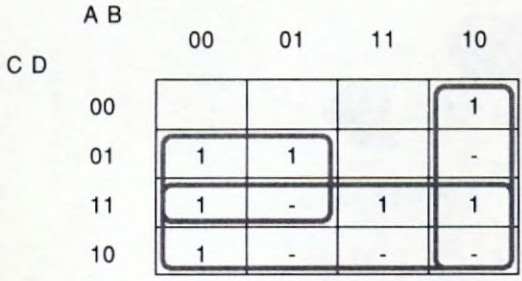
C \ A B	00	01	11	10
0		1	-	
1	1	1	-	1

$F(A,B,C)=B+C$

*## **Ejemplo 5.25**: Simplificar por Karnaugh la siguiente función (expresada mediante la tabla de verdad):*

A	B	C	D	F
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	-
0	1	1	1	-
1	0	0	0	1
1	0	0	1	-
1	0	1	0	-
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	-
1	1	1	1	1

Solución:



$$F=C+A\bar{B}+\bar{A}D$$

5.7. EXPRESION DE UNA FUNCION MEDIANTE PUERTAS LOGICAS

Una vez obtenida la expresión algebraica de una función mediante cualquiera de los métodos de simplificación descritos en los apartados previos, el último paso a realizar consiste en el diseño del circuito digital correspondiente a dicha función, empleando las puertas lógicas. Cualquier función se puede expresar mediante la agrupación y conexionado de las diferentes puertas lógicas existentes. De esta forma, se emplean normalmente numerosos tipos

de puertas lógicas, lo que en la realidad del montaje de dicho circuito equivale a tener que utilizar varios tipos de circuitos integrados, con el normal desaprovechamiento de los mismos, ya que en algunos casos se utiliza una sola puerta lógica cuando el circuito integrado puede contener cuatro ó seis de dichas puertas.

A veces se requiere una uniformidad en el empleo de puertas lógicas, es decir, utilizar un sólo modelo de puertas lógicas. Para este tipo de montajes se emplean las puertas lógicas NAND y NOR. Mediante un proceso de conversión de la expresión algebraica, se consigue que todos los términos que representan a una función puedan ser representados mediante dichas puertas, incluso las negaciones, que se obtienen conectando las dos entradas de una puerta NAND o NOR a la misma señal.

*## **Ejemplo 5.26**: Representar mediante puertas lógicas la expresión algebraica resultante de la simplificación del ejemplo 5.9.*

Solución:

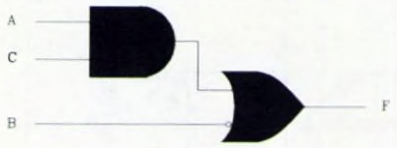


Figura 5.5

*## **Ejemplo 5.27**: Representar mediante puertas lógicas la expresión algebraica resultante de la simplificación del ejemplo 5.10.*

Solución:

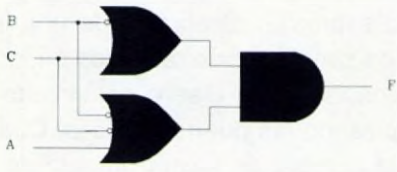


Figura 5.6

*## **Ejemplo 5.28**: Representar mediante puertas lógicas la expresión algebraica resultante de la simplificación del ejemplo 5.12.*

Solución:

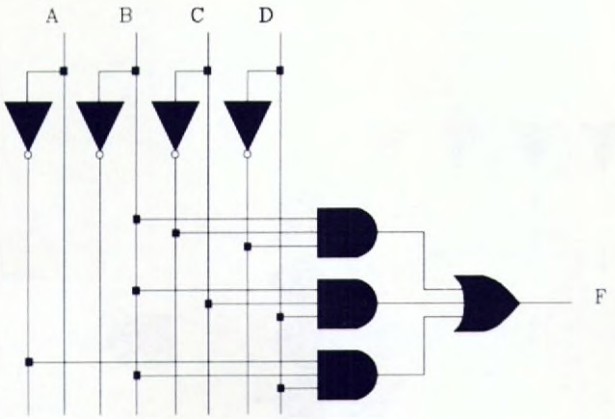


Figura 5.7

*## **Ejemplo 5.29**: Representar mediante puertas lógicas la expresión algebraica resultante de la simplificación del ejemplo 5.19.*

Solución:

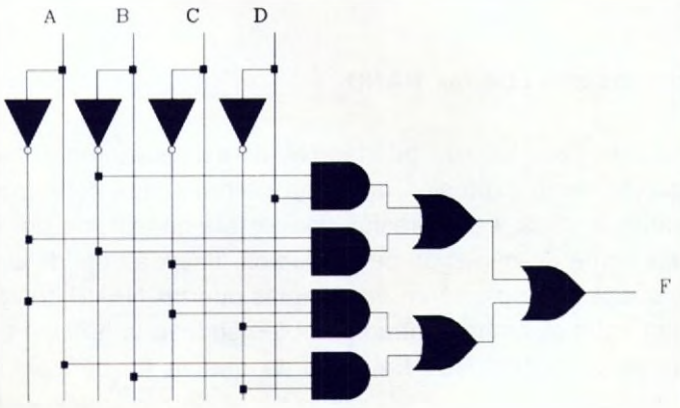


Figura 5.8

*## **Ejemplo 5.30**: Representar mediante puertas lógicas la expresión algebraica resultante de la simplificación del ejemplo 5.20.*

Solución:

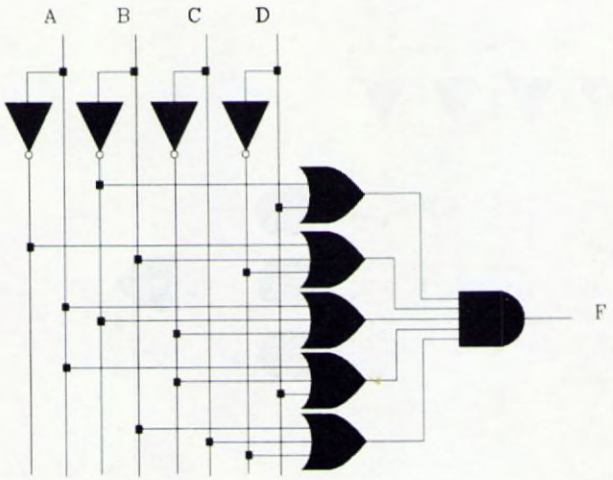


Figura 5.9

5.7.1. **Circuitos con puertas NAND**

Para obtener circuitos con puertas NAND exclusivamente, se realiza una doble negación de la expresión de suma de productos correspondiente a la función que se analiza. La expresión final estará constituida por una serie de términos en forma de negación de productos, después del desarrollo de una de las negaciones. Si se quieren emplear las puertas NAND como negadores, se conectan entre sí ambas entradas, obteniéndose la función equivalente a la que realiza la puerta NO. En caso de que la función empleada como desarrollo no estuviera expresada como suma de productos y existieran términos mixtos con productos y sumas, se realizará una doble negación parcial en dichos términos para unificar las expresiones.

*## **Ejemplo 5.31**: Obtener la representación mediante puertas NAND de la siguiente función:*

$$F(A,B,C)=AB\overline{C}+\overline{A}BC+A\overline{B}C$$

Solución:

$$F(A,B,C)=\overline{AB\overline{C}+\overline{A}BC+A\overline{B}C}=\overline{(AB\overline{C})(\overline{A}BC)(A\overline{B}C)}$$

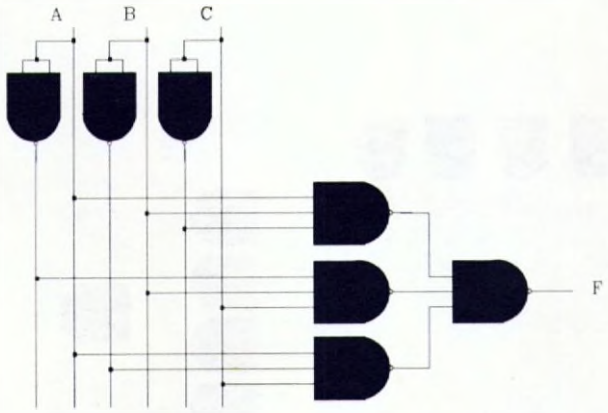


Figura 5.10

*## **Ejemplo 5.32**: Obtener la representación mediante puertas NAND de la función resultante de la simplificación del ejemplo 5.12.*

Solución:

$$F=\overline{(\overline{C}B\overline{D})(B\overline{C}D)(\overline{A}B\overline{D})}$$

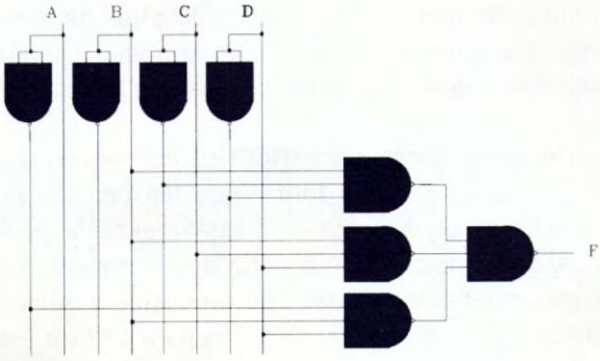


Figura 5.11

*## **Ejemplo 5.33**: Obtener la representación mediante puertas NAND de la función resultante de la simplificación del ejemplo 5.11.*

Solución:

$$F=(A+\overline{B})(B+\overline{D})(\overline{B}+\overline{C}+D)=AB\overline{C}+ABD+A\overline{C}\,\overline{D}+\overline{B}\,\overline{D}=$$

$$=\overline{AB\overline{C}+ABD+A\overline{C}\,\overline{D}+\overline{B}\,\overline{D}}=\overline{(AB\overline{C})}\,\overline{(ABD)}\,\overline{(A\overline{C}\,\overline{D})}\,\overline{(\overline{B}\,\overline{D})}$$

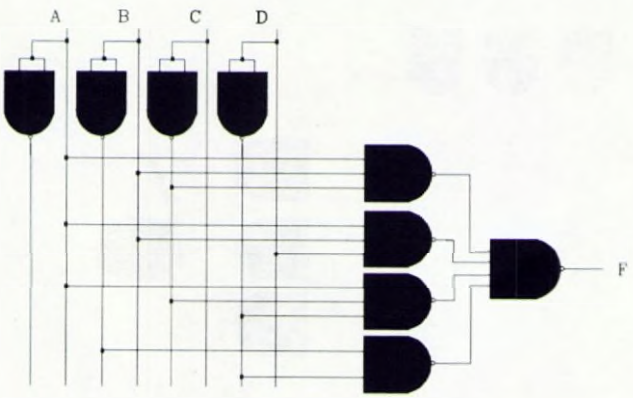


Figura 5.12

5.7.2. **Circuitos con puertas NOR**

Para obtener circuitos con puertas NOR exclusivamente, se realiza una cuádruple negación de la expresión de suma de productos correspondiente a la función que se analiza. La expresión final estará constituida por una serie de términos en forma de negación de sumas, después del desarrollo de tres de las negaciones. Después del desarrollo de la primera negación se deben realizar las operaciones algebraicas que simplifiquen la expresión. (Ver ejemplo 5.34).

Si se quieren emplear las puertas NOR como negadores, se conectan entre sí ambas entradas, de igual forma que las con las puertas NAND, obteniéndose la función equivalente a la que realiza la puerta NO. Si la función empleada como desarrollo no estuviera expresada como suma de productos y existieran términos mixtos con productos y sumas, se realizará una doble negación parcial en dichos términos para unificar las expresiones. Si está expresada la función como producto de sumas sólo se necesitan dos negaciones.

Existe un método más abreviado de realizar la expresión con puertas NOR que no es tan óptimo a veces como el indicado en los párrafos anteriores, pero que se emplea frecuentemente. Consiste en realizar tres negaciones de la expresión algebraica, lo que equivale a obtener la función negada. Conectando una puerta de negación al resultado final se obtiene el valor directo de la función (Equivale también a 4 negaciones). Se realiza desarrollando sólo la primera negación (Ver ejemplo 5.35). Dependiendo de la expresión algebraica, a veces este método resulta muy fácil aunque en ocasiones suele dar como resultado el empleo de mayor número de puertas lógicas.

*## **Ejemplo 5.34**: Obtener la representación mediante puertas NOR de la función siguiente:*

$$F(A,B,C)=AB+\overline{A}BC+A\overline{B}C$$

Solución:

$$F(A,B,C)=\overline{\overline{AB}+\overline{ABC}+\overline{A\overline{B}C}}=\overline{(\overline{A}+\overline{B})}\,\overline{(A+\overline{B}+\overline{C})}\,\overline{(\overline{A}+B+\overline{C})}=$$

$$=\overline{\overline{\overline{A}\overline{B}}+\overline{\overline{A}\overline{C}}+\overline{B\overline{C}}}=\overline{(A+B)(A+C)(B+C)}=\overline{(A+B)}+\overline{(A+C)}+\overline{(B+C)}$$

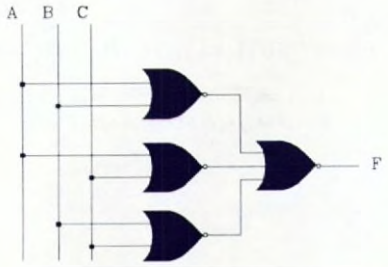


Figura 5.13

*## **Ejemplo 5.35**: Obtener la representación mediante puertas NOR de la función del problema 5.34, empleando el método abreviado.*

Solución:

$$\overline{F}=\overline{AB+\overline{A}BC+A\overline{B}C}=\overline{(A+B)}\,\overline{(A+\overline{B}+C)}\,\overline{(\overline{A}+B+C)}=\overline{(A+B)}+\overline{(A+\overline{B}+C)}+\overline{(\overline{A}+B+C)}$$

$$F(A,B,C)=\overline{\overline{(A+B)}+\overline{(A+\overline{B}+C)}+\overline{(\overline{A}+B+C)}}$$

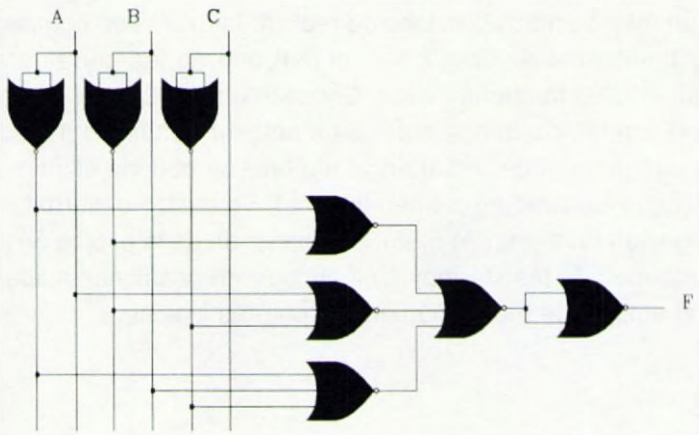


Figura 5.14

*## **Ejemplo 5.36**: Representar mediante puertas NOR el circuito del ejemplo 5.16 utilizando el método abreviado.*

Solución:

$$F=\overline{\overline{B}C+\overline{B}D+A\overline{B}\overline{C}\overline{D}}=\overline{(B+\overline{C})(B+\overline{D})(A+\overline{B}+C+D)}=\overline{(B+\overline{C})}+\overline{(B+\overline{D})}+\overline{(A+\overline{B}+C+D)}$$

$$F=(B+\overline{C})+(B+\overline{D})+(A+\overline{B}+C+D)$$

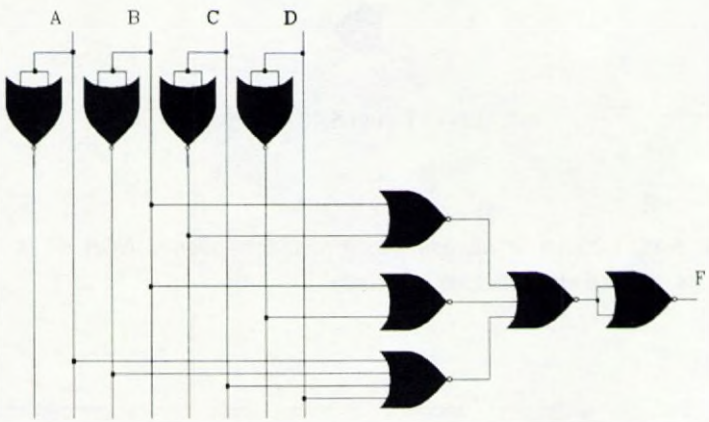


Figura 5.15

5.8. **EJEMPLOS**

*## **Ejemplo 5.37**: Simplificar la siguiente función algebraica de 4 variables mediante álgebra de Boole, obtener la tabla de verdad, simplificar por Karnaugh empleando suma de productos y representar el circuito con puertas lógicas:*

$$F=\overline{A}BCD+\overline{ABCD+\overline{A}B\overline{C}D}+\overline{B}C\overline{D}+\overline{A}C\overline{D}$$

Solución:

Simplificación algebraica:

$$F=\overline{A}BCD+(\overline{A}+\overline{B}+\overline{C}+\overline{D})(A+\overline{B}+C+\overline{D})+\overline{B}C\overline{D}+\overline{A}C\overline{D}=$$

$$=\overline{A}BCD+\overline{B}+\overline{A}C+A\overline{C}+\overline{D}+\overline{B}C\overline{D}+\overline{A}C\overline{D}=$$

$$=\overline{B}+\overline{A}C+A\overline{C}+\overline{D}$$

Tabla de verdad:

A	B	C	D	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Simplificación por Karnaugh:

$$F=m_0+m_1+m_2+m_3+m_4+m_6+m_7+m_8+m_9+m_{10}+m_{11}+m_{12}+m_{13}+m_{14}$$

C D \ A B	00	01	11	10
00	1	1	1	1
01	1		1	1
11	1	1		1
10	1	1	1	1

Ecuación algebraica:

$$F=\bar{A}+\bar{D}+\bar{A}C+A\bar{C}$$

Representación gráfica del circuito:

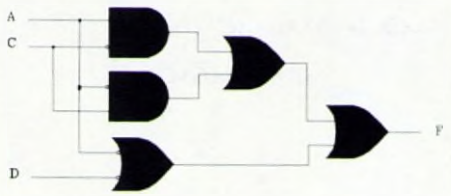


Figura 5.16

*## **Ejemplo 5.38**: Simplificar la siguiente función de 4 variables mediante las 4 posibles opciones de Karnaugh y Quine McCluskey y representar el circuito con puertas NAND:*

$$F=m_0+m_1+m_2+m_4+m_7+m_8+m_9$$

Solución:

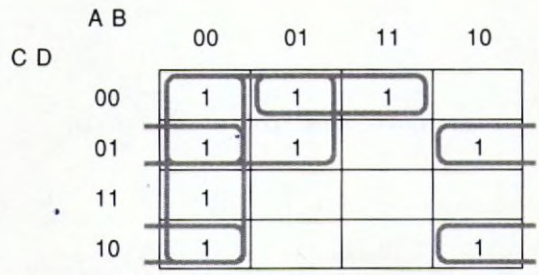
S-1) Karnaugh usando suma de productos:

C D \ A B	00	01	11	10
00	1	1		1
01	1			1
11		1		
10	1			

$$F=\bar{B}\bar{C}+\bar{A}C\bar{D}+\bar{A}\bar{B}\bar{D}+\bar{A}BCD$$

S-2) Karnaugh usando producto de sumas:

$$F=M_0M_1M_2M_3M_4M_5M_9M_{10}M_{12}$$



$$F=(\bar{A}+\bar{B})(\bar{A}+\bar{C})(B+\bar{C}+\bar{D})(\bar{B}+\bar{C}+D)(\bar{B}+C+\bar{D})$$

S-3) Quine McCluskey usando suma de productos:

0000	000-	-00-
------	00-0	
0001	0-00	
0010	-000	
0100	------	
1000	-001	
------	100-	
1001		

0111		

	0000	**0001**	**0010**	**0100**	**1000**	**1001**	**0111**
-00-	x	x			x	x	
00-0	x		x				
0-00	x			x			
0111							x

$$F=\bar{B}\bar{C}+\bar{A}\bar{B}\bar{D}+\bar{A}\bar{C}\bar{D}+\bar{A}BCD$$

S-4) Quine McCluskey usando producto de sumas:

0000	000-	00--
------	00-0	0-0-
0001	0-00	
0010	------	
0100	00-1	
------	0-01	
0011	-001	
0101	001-	
1001	-010	
1010	010-	
1100	-100	

	0000	0001	0010	0100	0011	0101	1001	1010	1100
00--	x	x	x		x				
0-0-	x	x		x		x			
-100				x					x
-010			x					x	
-001		x					x		

$$F=(\bar{A}+\bar{B})(\bar{A}+\bar{C})(B+\bar{C}+\bar{D})(\bar{B}+C+\bar{D})(\bar{B}+\bar{C}+D)$$

Representación gráfica del circuito:

$$\overline{F=\bar{B}\bar{C}+A\bar{B}\bar{D}+\bar{A}\bar{C}\bar{D}+\bar{A}BCD}=\overline{(\bar{B}\bar{C})(A\bar{B}\bar{D})(\bar{A}\bar{C}\bar{D})(\bar{A}BCD)}$$

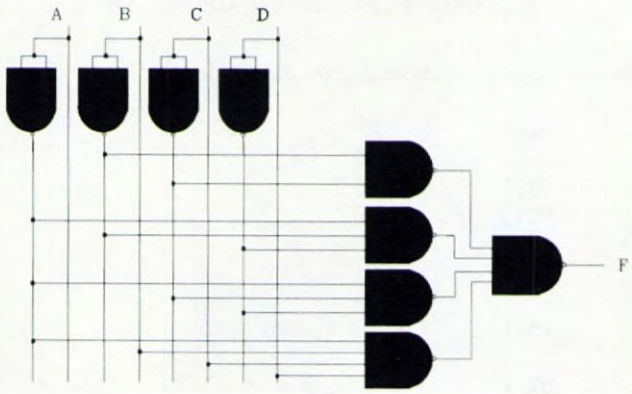


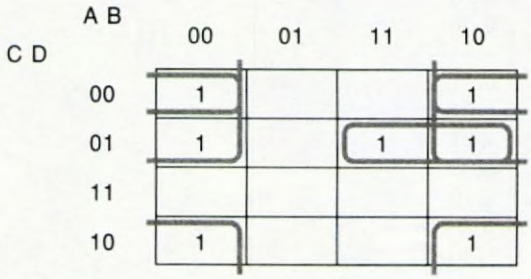
Figura 5.17

*## **Ejemplo 5.39**: Simplificar la siguiente función de 4 variables mediante las 4 posibles opciones de Karnaugh y Quine McCluskey y representar el circuito con puertas NOR:*

$$F=m_0+m_1+m_2+m_8+m_9+m_{10}+m_{13}$$

Solución:

S-1) Karnaugh usando suma de productos:



$$F=\bar{B}\bar{D}+\bar{B}\bar{C}+A\bar{C}\bar{D}$$

S-2) Karnaugh usando producto de sumas:

$$F=M_0M_1M_3M_4M_8M_9M_{10}M_{11}M_{12}$$

C D \ A B	00	01	11	10
00	1	1	1	1
01	1			1
11	1			1
10				1

$$F=(A+\bar{B})(\bar{C}+\bar{D})(\bar{B}+D)$$

S-3) Quine McCluskey usando suma de productos:

0000	000-	-00-
------	00-0	-0-0
0001	-000	
0010	------	
1000	-001	
------	-010	
1001	100-	
1010	10-0	
------	------	
1101	1-01	

	0000	**0001**	**0010**	**1000**	**1001**	**1010**	**1101**
-00-	x	x		x	x		
-0-0	x		x	x		x	
1-01					x		x

$$F=\bar{B}\bar{D}+\bar{B}\bar{C}+A\bar{C}D$$

S-4) Quine McCluskey usando producto de sumas:

0000	000-	-00-
------	0-00	--00
0001	-000	------
0100	------	-0-1
1000	00-1	10--
------	-001	
0011	-100	
1001	100-	
1010	10-0	
1100	1-00	
------	------	
1011	-011	
	10-1	
	101-	

	0000	0001	0100	1000	0011	1001	1010	1100	1011
-00-	x	x		x		x			
--00	x		x	x				x	
-0-1		x			x	x			x
10--				x		x	x		x

$$F=(\overline{C}+\overline{D})(\overline{B}+D)(A+\overline{B})$$

Representación gráfica del circuito:

$$F=\overline{\overline{(\overline{C}+\overline{D})(\overline{B}+D)(A+\overline{B})}}=\overline{\overline{(\overline{C}+\overline{D})}+\overline{(\overline{B}+D)}+(A+\overline{B})}$$

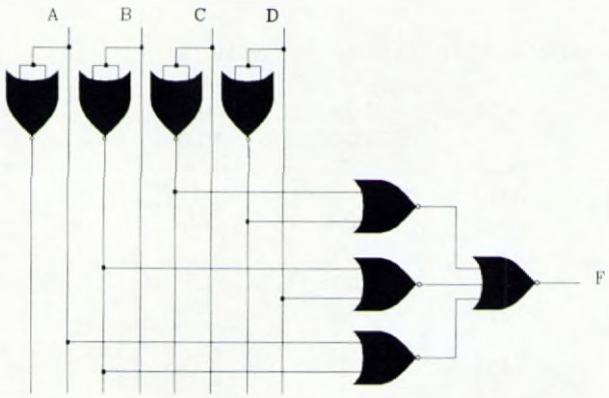


Figura 5.18

*## **Ejemplo 5.40**: Obtener la ecuación algebraica de definición, la tabla de verdad y las expresiones de suma de productos y producto de sumas del esquema bloque representado en la figura 5.19. sabiendo que dicho circuito responde al cronograma de la figura 5.20.*

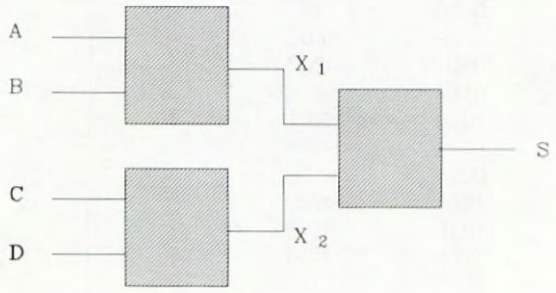


Figura 5.19

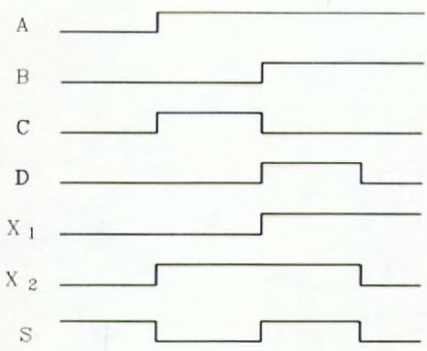


Figura 5.20

Solución:

Ecuación algebraica:

$$S=X_1 \odot X_2$$
$$X_1=AB$$
$$X_2=C+D$$

$$S=AB\odot(C+D)=\overline{AB}(\overline{C+D})+AB(C+D)=\bar{A}\bar{C}\bar{D}+\bar{B}\bar{C}\bar{D}+ABC+ABD$$

Tabla de verdad:

A	B	C	D	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Expresión de suma de productos:

$$F(A,B,C,D)=m_0+m_4+m_8+m_{13}+m_{14}+m_{15}$$

Expresión de producto de sumas:

$$F(A,B,C,D)=M_3M_4M_5M_6M_8M_9M_{10}M_{12}M_{13}M_{14}$$

TEMA 6
CIRCUITOS INTEGRADOS

6

CIRCUITOS INTEGRADOS

6.1. INTRODUCCION

Se describen a continuación las características más importantes de los circuitos integrados, tanto desde el punto de vista de su fabricación como del posterior uso en equipos digitales. Se analiza el proceso de fabricación de los distintos tipos de circuitos, las escalas de integración y los componentes que encierra cada uno de los elementos físicos que se manipula en la fabricación y diseño de equipos electrónicos.

Se tienen en cuenta las ventajas e inconvenientes de cada una de las familias y tecnologías en que se han estructurado los circuitos existentes en el mercado, así como los parámetros y características técnicas de entrada, salida, tensión, corriente, tiempo, etc. de cada uno de los modelos. La referencia a las nomenclaturas empleadas por cada fabricante, así como los esquemas representativos de cada configuración externa y el contenido y conexionado interno harán más fácil y comprensible los conceptos teóricos introducidos en temas anteriores y posteriores.

6.2. EL CIRCUITO INTEGRADO

Un circuito integrado es un circuito electrónico miniaturizado, formado por transistores, diodos, resistencias, condensadores, etc., construidos mediante la más moderna tecnología. Necesitan una tensión pequeña de alimentación, y sus señales de salida dependen de las señales de entrada, aplicadas y obtenidas a través de los terminales exteriores del encapsulado del C.I.

Se denomina lógica positiva a la asignación de valor lógico "1" al nivel de tensión alto y "0" al nivel de tensión bajo como ya se ha visto anteriormente. La lógica negativa toma los valores opuestos, existiendo fabricantes y productos para ambos tipos de lógica.

6.3. VENTAJAS E INCONVENIENTES DE LOS CIRCUITOS INTEGRADOS

En relación con otros dispositivos electrónicos, los circuitos integrados tienen unos costes muy bajos, debido al volumen tan elevado de componentes que se fabrican, al menor número de elementos internos, menor cantidad de materiales empleados, y al tamaño tan reducido de los mismos. Su fiabilidad es muy elevada debido a los procesos de fabricación tan estrictos, a los controles unitarios existentes en la producción, al encapsulado que aisla los componentes de agentes externos, y a la distribución uniforme de temperatura gracias al tamaño y al encapsulado.

Permiten obtener una clara mejora en los resultados, dado que al eliminarse el cableado, las conexiones, soldaduras, etc., la respuesta del integrado es mucho más rápida y precisa que la de otros componentes. Sus reparaciones son más simples y económicas, ya que la sustitución de un componente es suficiente para resolver el problema causado, con el consiguiente ahorro en materiales y mano de obra. La eliminación de fallos de montaje, soldadura, errores humanos, manipulación, etc., contribuye a consolidar firmemente las enormes ventajas y posibilidades de estos circuitos.

Los pocos inconvenientes que tienen estos elementos están centrados en la gran dificultad de integrar resistencias y condensadores de valores elevados, dada la limitación térmica del circuito integrado. Esta limitación térmica, hace que la potencia disipada no pueda ser elevada por el riesgo de destrucción. No es posible integrar inductancias, aunque en la actualidad se consiguen efectos equivalentes, y necesitan herramientas especiales para ser manipulados sin deterioro.

6.4. METODO DE FABRICACION

Según la tecnología y las distintas familias de C.I. pueden existir diversos procesos de fabricación. El proceso genérico comienza con la obtención de las obleas de silicio tipo N ó P, que servirán de sustrato base para cientos de C.I. Mediante oxidación en hornos de alta temperatura y atmósfera gaseosa controlada, se consiguen zonas semiconductoras tipo P ó N, dependiendo del material de la oblea base.

Estas zonas N y P, pueden combinarse formando transistores, diodos, resistencias, etc., mediante máscaras que dejarán al descubierto las zonas donde tendrán que inyectarse los correspondientes portadores de cada material. El proceso continúa con recubrimientos antioxidantes y aislantes de las distintas capas que se van formando. Mediante técnicas de fotograbado y exposición a luz ultravioleta, se eliminan las capas de óxido y de barniz aislante.

También mediante proceso de fotograbado se recubren con una capa de material conductor (normalmente aluminio) las zonas de conexiones. Una vez terminado el C.I. se inspeccionan unitariamente y se identifican y marcan los defectuosos,

pasando posteriormente la oblea al corte de todos los circuitos fabricados. Posteriormente, se soldarán los terminales con hilo de aluminio, platino, oro, o cualquier otro material conductor, y se realizará el montaje y encapsulado. Todo el proceso es controlado automáticamente por ordenadores.

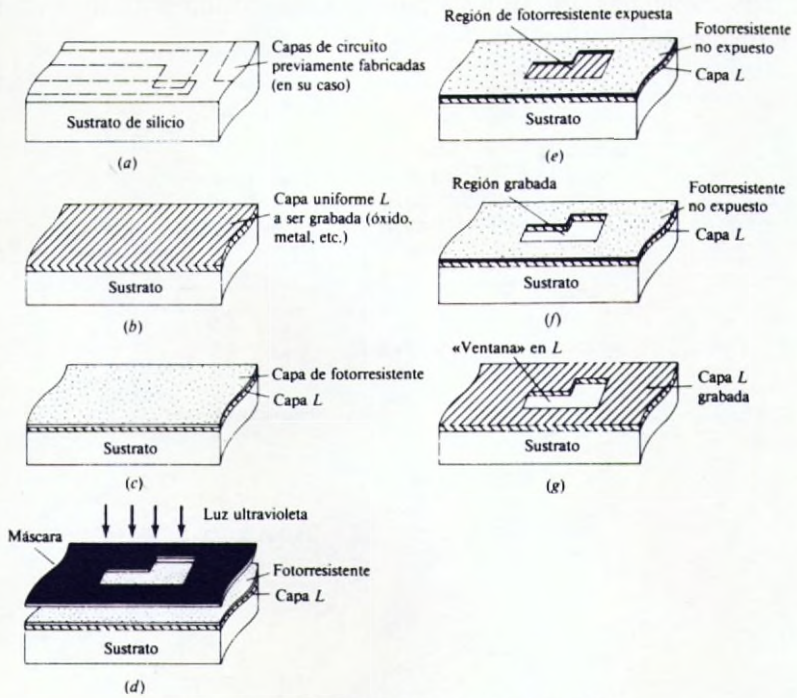


Figura 6.1. Proceso de fabricación de un C.I.

6.5. **ENCAPSULADOS**

Según las aplicaciones de los diversos circuitos integrados, se pueden encapsular de formas diferentes, que tienen que ver tanto con el número de elementos que existen en su interior, como con la utilización de los mismos y la disipación térmica que se produzca. En las figuras 6.2 se pueden ver los esquemas correspondientes a las configuraciones más importantes. En dichos esquemas se hace referencia a la numeración de los terminales y al punto de referencia que sirve de comienzo de la cuenta de dichos terminales, existiendo encapsulados metálicos, de material plástico y cerámicos.

* Cápsula cilíndrica: Es de metal, de forma parecida a los transistores. El máximo número de terminales exteriores es 10, y tienen aplicaciones muy limitadas siendo su principal uso para el montaje de amplificadores operacionales.

* Cápsula plana: Se denomina también flat-pack. Es de material cerámico, aunque en algunos casos se construyen también en material plástico. Está especialmente preparado para el montaje en máquinas con soldadura automática o semiautomática por puntos lo que hace muy difícil su uso por medios normales. El número de conexiones exteriores es variable, siendo los más habituales 14, 16 y 24 pines, pudiendo tomar diferentes ángulos de terminación.

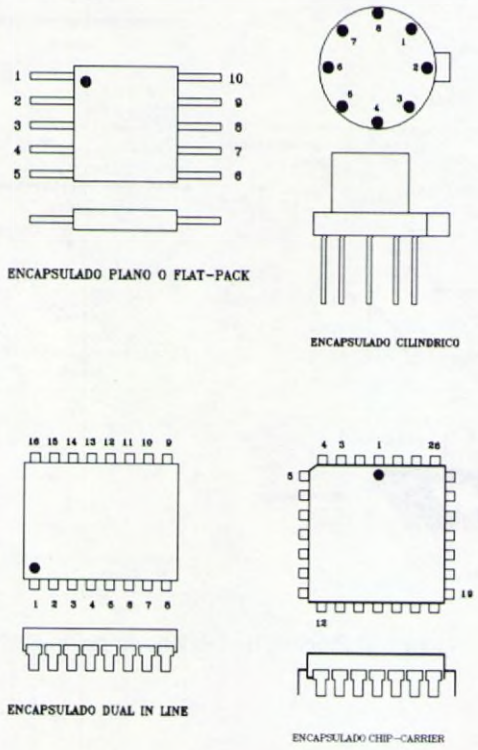


Figura 6.2. Encapsulados de circuitos integrados

* Dual in line: Abreviado DIP ó DIL. Está formado por 2 filas de conexiones exteriores paralelas. Puede decirse que es el encapsulado genérico y más representativo de los circuitos integrados. Puede ser de material plástico o cerámico. Suele acoplarse a zócalos previamente soldados al circuito impreso, lo que facilita su desmontaje y sustitución. No tiene una limitación de terminales, fabricándose de 14, 16, 32 pines, etc.

* Chip carrier: Es un circuito de forma cuadrada con terminales en las cuatro caras laterales. Pueden ser de material plástico o cerámico. Suelen ser

opciones especiales de circuitos "dual in line" aunque se está extendiendo mucho en fabricación de microprocesadores y memorias.

6.6. ESCALAS DE INTEGRACION

Dependiendo del número de transistores o puertas lógicas que contienen interiormente los circuitos integrados, se pueden formar cuatro grupos de integración, que están asimismo relacionados con las distintas tecnologías o familias lógicas existentes:

* SSI (Small Scale Integration): Son circuitos integrados con funciones elementales, conteniendo entre 1 y 12 puertas lógicas, de aplicación general, constituyendo especialmente los elementos más simples de la electrónica digital, como por ejemplo las puertas NAND, NOR, OR, etc.

* MSI (Medium Scale Integration): Son circuitos integrados con funciones más complejas, conteniendo entre 13 y 99 puertas lógicas, de aplicación general, como por ejemplo en contadores, decodificadores, registros de desplazamiento, etc.

* LSI (Large Scale Integration):Son circuitos integrados con funciones muy complejas, conteniendo entre 100 y 1000 puertas, de aplicación específica, como por ejemplo C.I. para calculadoras, para microprocesadores, memorias, etc.

* VLSI (Very Large Scale Integration): Circuitos integrados conteniendo más de 1000 puertas, de aplicación específica, especialmente realizados bajo diseño particular de determinados productos y fabricantes, como por ejemplo en grandes equipos informáticos, procesos de fabricación, robótica, etc.

6.7. FAMILIAS LOGICAS DE CIRCUITOS INTEGRADOS

Dependiendo de los elementos constitutivos de los circuitos integrados y de las diferentes tecnologías de fabricación de los mismos, se han ido desarrollando las denominadas familias lógicas, que engloban circuitos de similares características. En función de dichas características, se pueden reseñar las familias lógicas y tecnologías que a continuación se detallan, como las más relevantes.

6.7.1. **Familia lógica RTL**

Es la familia lógica más antigua que se ha fabricado, aunque ya no tiene apenas uso. Las siglas corresponden a LOGICA RESISTENCIA-TRANSISTOR. El circuito básico es una puerta NOR con un tiempo de propagación de 12 ns, y consumo de 10 mw por puerta, ofreciendo un fanout de 5 puertas y baja inmunidad al ruido. Su alta densidad de integración repercute en un bajo coste. La tensión de alimentación puede oscilar entre 3 y 3.5 V.

6.7.2. **Familia lógica DTL**

Es también una de las más antiguas, siendo su circuito básico una puerta NAND. Las siglas corresponden a LOGICA DIODO-TRANSISTOR. El tiempo de propagación es de 30 ns, con un fanout de 5 puertas, baja inmunidad al ruido, tensión de alimentación de 5 V., y consumo de 10 mw por puerta.

6.7.3. **Familia lógica HTL**

Constituyen los C.I. de alta inmunidad al ruido. De parecido diseño a la familia lógica DTL, se usan en cambios continuos de tensión, con interruptores, tiristores, líneas de conducción, telefonía, etc., siendo su circuito básico una puerta NAND. Su velocidad de propagación es muy lenta, aproximadamente 150 ns, con una tensión de alimentación de 15 V, un fanout de 10 y consumo de 55 mw por puerta. Tiene una subdivisión llamada HLL (High Level Logic), caracterizada por una tensión de alimentación entre 10 y 20 V, velocidad de propagación de 110 ns, fanout de 25 y consumo 20 mw por puerta.

6.7.4. **Familia lógica TTL**

Siendo la configuración más empleada en circuitos integrados, se engloban principalmente en escalas de integración SSI y MSI. Corresponden a LOGICA TRANSISTOR-TRANSISTOR. Actualmente, se están consiguiendo continuas mejoras dado el bajo coste y la gran producción de los mismos. Existen varios subgrupos dentro de esta familia lógica en función de las características de los mismos:

* TTL Standard: El circuito básico es una puerta NAND. La tensión de alimentación es de 5 V, velocidad de transmisión 10 ns, consumo 10 mw por

puerta, es de alta inmunidad al ruido y tiene un fanout de 10. Es el modelo de mayor fabricación.

* TTL de baja potencia (LPTTL): El circuito bàsico es una puerta NAND. El consumo es menor que en la serie stándard ya que se reduce la corriente debido al aumento de la resistencias. La velocidad de transmisión es de 33 ns, el consumo 1 mw por puerta, la tensión de alimentación 5 V, fanout 10 y es de alta inmunidad al ruido. Se emplea para bajo consumo y mínima disipación.

* TTL de alta velocidad (HTTL): El circuíto básico es una puerta NAND. Aumenta el número de transistores y disminuyen las resistencias. La velocidad de transmisión es de 6 ns, el consumo 22 mw por puerta, fanout 10, tensión de alimentación 5 V y menor inmunidad al ruido que la serie stándard.

* TTL Schottky: El circuíto básico es una puerta NAND. Está constituído por diodos Schottky lo que le convierte en el TTL más rápido. El proceso de fabricación es el más simple de toda la serie TTL. La velocidad de transmisión es de 3 ns, el consumo 19 mw por puerta, fanout 10, tensión de alimentación 5 V y alta inmunidad del ruido.

* TTL Schottky de baja potencia (LSTTL): El circuíto básico es una puerta NAND. Es el mismo tipo de circuito que la serie TTL Schottky, pero aumentan los valores de las resistencias. Posee menor inmunidad al ruido que el anterior, una velocidad de transmisión de 10 ns, consumo 2 mw por puerta, fanout 10 y tensión de alimentación 5 V.

6.7.5. **Tecnología MOS**

Los transistores de efecto de campo (FET) se diferencian de los semiconductores normales en que solo tienen una de las zonas formadas por material semiconductor. La composición de estos transistores se abrevia con el término MOS que significa metal-oxido-semiconductor. Basado en esta composición y dependiendo del tipo de sustrato semiconductor, se pueden obtener diversos modelos de transistores y por tanto de circuitos integrados. Este tipo de circuitos se emplea fundamentalmente en escalas de integración SSI y MSI.

* Circuitos integrados CMOS: El elemento básico es el inversor, aunque los circuitos básicos pueden ser puertas NOR o NAND. La tensión de alimen-

tación oscila entre 3 y 16 V. para la serie stándar y 3 y 18 V. para la serie especial, influyendo directamente en la velocidad de conmutación. El consumo oscila entre 0.01 y 10 mw. por puerta. Poseen gran inmunidad al ruido, un mínimo fanout de 50 y velocidad de transmisión 70 ns.

* Circuitos integrados HCMOS: Son circuitos CMOS de alta velocidad. La tensión de alimentación oscila entre 2 y 6 V, el consumo es de 2.5 mw por puerta y la velocidad de transmisión es de 9 ns. Existe una serie compatible con la TTL con alimentación de 5 V.

6.7.6. **Tecnología ECL**

Denominado Emitter Coupled Logic, posee circuitos integrados de muy alta velocidad, empleados en escala de integración SSI. Dispone de salida complementaria, por lo que en el mismo circuito básico se encuentran las puertas OR y las puertas NOR (modelo básico). Es un C.I. de gran producción, especialmente empleado en sistemas de comunicaciones de alta velocidad, satélites, etc.

La tensión de alimentación es negativa, -5 V, mostrando un abanico de salida de tensión desde valores negativos hasta positivos. El fanout es alto, de 25 puertas, el consumo es de 25 mw por puerta y poseen baja inmunidad al ruido. Existen cuatro series dentro de la familia, en función de la velocidad de transmisión (8, 4, 2 y 1 ns). La serie más popular es la ECL de 2 ns, que combina la velocidad y el consumo óptimos. Para las series de 1 y 2 ns se necesitan circuitos impresos de nueva generación, dado que pueden llegar a excitarse corrientes en líneas paralelas del circuito impreso por la alta velocidad de transmisión de las señales.

6.7.7. **Tecnología IIL**

Denominada también I2L ó Integrated Injection Logic, es una de las familias de aparición más recientes y una de las que cuenta con más futuro, empleándose en escalas de integración LSI y VLSI. Mejora a la serie NMOS en velocidad y a la CMOS en consumo, pudiendo su densidad de integración superar ampliamente las 300 puertas por mm^2.

Al igual que la ECL, el circuito básico presenta salida complementaria, por lo que se obtienen las puertas OR y NOR simultáneamente. La tecnología de fabricación es muy simplificada al tener menos contactos, no existir resistencias, no disponer de cableado entre puertas y llevar diodos Schottky. Son

importantes fundamentalmente por su alta velocidad y bajo consumo, siendo la tensión de alimentación del orden de 1 V., y se emplean fundamentalmente en telecomunicación.

6.7.8. **Otras familias lógicas**

Aunque en menor escala de importancia en cuanto a fabricación actual, pero con un enorme potencial futuro, se encuentran un grupo de familias circunscritas a las escalas de integración LSI y VLSI. Con ellas se trata de obtener mayor velocidad, menor consumo y mayor densidad de integración. Entre ellas se pueden citar las siguientes: NMOS, PMOS, VMOS, HMOS, DMOS, SOS (Silicon on Sapphire), IIIL (I3L) y JIL.

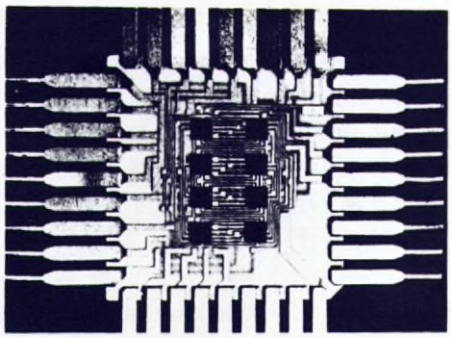


Figura 6.3. Circuito integrado (Interior del encapsulado)

6.8. **CARACTERISTICAS DE LOS CIRCUITOS INTEGRADOS. PARAMETROS**

Se pueden definir las características de un circuito integrado mediante la descripción de sus parámetros identificativos. Dichos parámetros, definen al mismo tiempo algunas de las características de las puertas lógicas que constituyen el interior del circuito integrado.

* Datos de identificación: En el exterior de los circuitos integrados están grabadas las características básicas que lo definen además de las marcas

visuales de identificación del terminal número 1. Dependiente de la tecnología y del fabricante, existen los siguientes campos genéricos:

- Nombre del fabricante: Mediante símbolos o siglas
- Prefijo: Producto estandar o particular de fabricante
- Referencia del circuito: Serie y clase de circuito
- Encapsulado: Referencia de encapsulado del fabricante

Por ejemplo el circuito TI-SN-74LS01-J es de Texas Instrument (TI), modelo estandar (SN), serie 74, familia LSTTL, modelo de cuatro puertas NAND de 2 entradas (01), encapsulado DIL de material cerámico (J).

* Velocidad de transmisión: Es el tiempo de respuesta de una puerta lógica desde el instante en que se aplica la señal y el momento en que la salida adopta el valor lógico correspondiente a dicha entrada. Esta característica va íntimamente relacionada con el tiempo de retardo en la propagación (propagation delay), que es el tiempo que tarda en aparecer en la salida la señal correspondiente a la entrada aplicada. En la familia TTL por ejemplo, se calcula el retardo en la propagación a 1.5 V tanto para el paso a nivel alto (t_{PLH}) como para el bajo (t_{PHL}).

* Tiempo de subida (rise time): Tiempo que tarda la señal en pasar del 10 al 90% de la tensión. Los valores lógicos "0" y "1" se suelen representar como "L" (nivel bajo ó LOW) y "H" (nivel alto ó HIGH).

* Tiempo de bajada (fall time): Tiempo que tarda la señal en pasar del 90 al 10% de la tensión.

* Potencia de disipación: Potencia consumida por el total de las puertas lógicas de un C.I., expresado normalmente en el consumo por puerta lógica (mw).

* Fan-in: Número máximo de puertas de la misma familia que se pueden conectar a la entrada de una puerta lógica sin exceder los valores máximos de corriente permitidos.

* Fan-out: Número máximo de puertas de la misma familia que se pueden conectar a la salida de una puerta lógica en conducción sin superar los valores máximos de corriente permitidos.

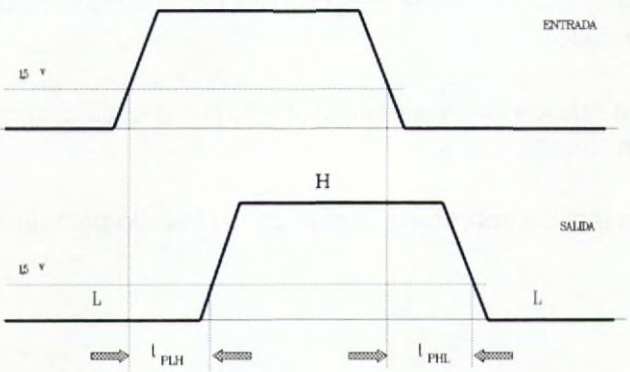


Figura 6.4. Tiempos de retardo en la propagación

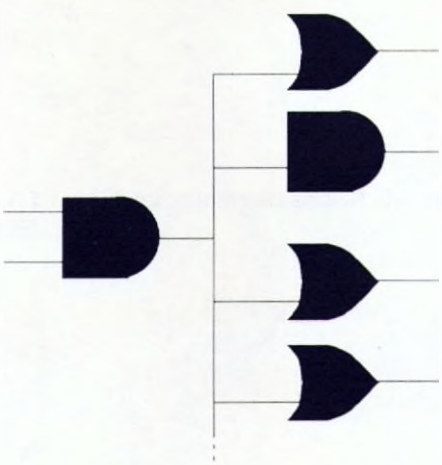


Figura 6.5. Fan-out

* Inmunidad al ruido: Es la máxima variación de la señal de entrada debido a perturbaciones sin que afecte a la señal de salida del circuito. La señal de ruido se puede acoplar o superponer a las señales de entrada y salida, transmitiéndose en cascada si existen varias puertas. Dependiendo de las familias los márgenes pueden variar. En la figura 6.6 se representan los valores de tensión para la familia TTL y en la figura 6.7 para la familia CMOS. Los márgenes de ruido (noise margin) suministrados por los fabricantes marcan la diferencia entre los valores garantizados para la salida y los aceptados para las entradas. Estos valores tienen los siguientes significados:

V_{OH}: Mínima tensión de salida suministrada por el circuito cuando su salida está en "alto".

V_{OL}: Máxima tensión de salida suministrada por el circuito cuando su salida está en "bajo".

V_{IH}: Mínima tensión aplicable a la entrada del circuito para que la salida sea "alta".

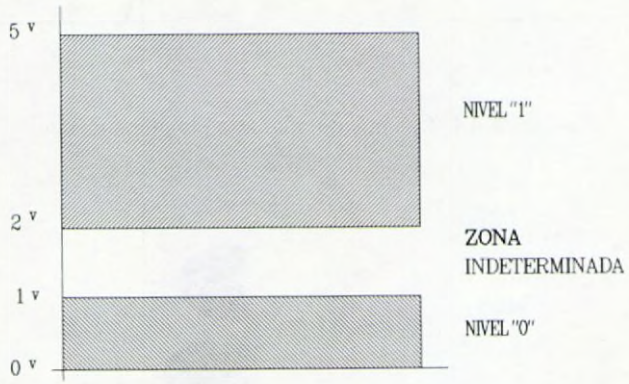


Figura 6.6. Niveles de entrada/salida para TTL

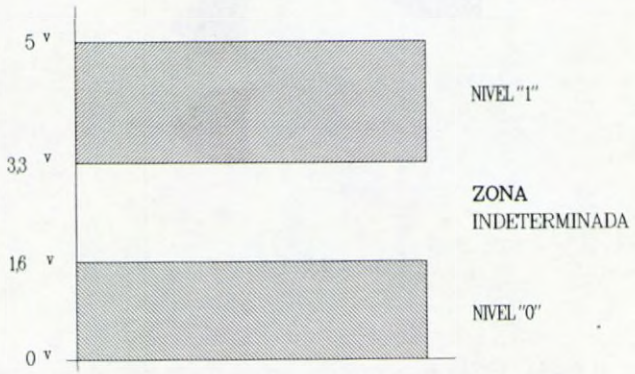


Figura 6.7. Niveles de entrada/salida para CMOS

V_{IL}: Máxima tensión aplicable a la entrada del circuito para que la salida sea "baja".

Las corrientes de entrada y salida que se relacionan con las tensiones de entrada y salida descritas, se denominan I_{OH}, I_{OL}, I_{IH} e I_{IL}.

* Características de transferencia: Es la relación existente entre las tensiones y corrientes de entrada y salida, expresada normalmente en forma de gráfica. Mediante rectas de carga se analizan las características de los circuitos.

* Interconexión (interfacing): Es la posibilidad de conectar circuitos integrados de distintas familias. Para que esto pueda realizarse deben ser compatibles los márgenes de tensión alto y bajo de ambos circuitos.

*## **Ejemplo 6.1**: Representar gráficamente las señales reales de entrada y salida en una puerta NO de lógica TTL, 5V, con $t_{PLH}=20^{ns}$ y $t_{PHL}=12^{ns}$.*

Figura 6.8

Solución:

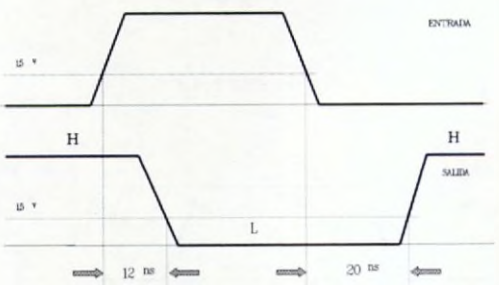


Figura 6.9

*## **Ejemplo 6.2**: Representar gráficamente las señales reales de entrada y salida en una puerta NOR de lógica TTL con una entrada conectada a masa y la otra a un pulso, teniendo el circuito como características 5V, $t_{PLH}=22^{ns}$ y $t_{PHL}=12^{ns}$.*

Figura 6.10

Solución:

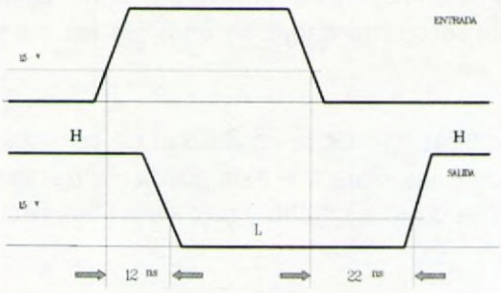


Figura 6.11

*## **Ejemplo 6.3**: Representar gráficamente las señales reales de entrada y salida en una puerta AND de lógica TTL con una entrada conectada a un "1" y la otra a un pulso, teniendo el circuito como características 5V, $t_{PLH}=18^{ns}$ y $t_{PHL}=20^{ns}$.*

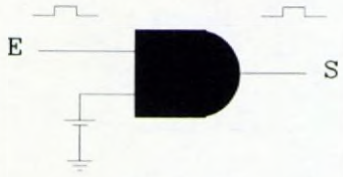


Figura 6.12

Solución:

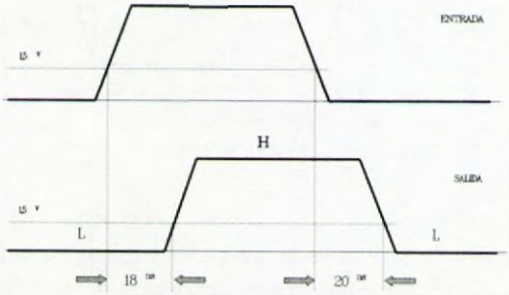


Figura 6.13

*## **Ejemplo 6.4**: Calcular el período, la frecuencia, los tiempos de subida y bajada y los tiempos de retardo en la propagación (subida y bajada) de una puerta XNOR con una entrada conectada a un "1" y la otra conectada a una señal cuyas características junto con las de salida se representan en la figura 6.15 (Tiempos en nanosegundos).*

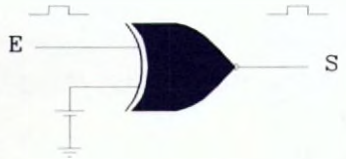


Figura 6.14

Solución:

Período: $T=100+120=220^{ns}$

Frecuencia: $F=1/T=1/220^{ns}=1/0,000000220^{s}=4545454,545^{Hz}=4,545^{MHz}$

Tiempo de subida: $t_s=9^{ns}$

Tiempo de bajada: $t_b=8^{ns}$

Tiempo de retardo en la propagación:

$t_{PLH}=15^{ns}$

$t_{PHL}=17^{ns}$

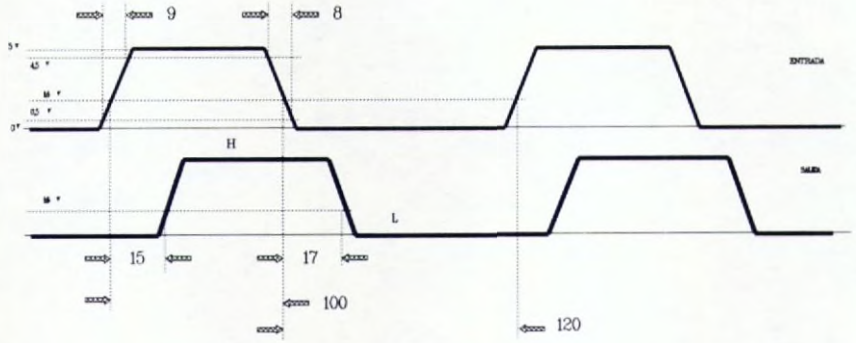


Figura 6.15

***## Ejemplo 6.5**: Calcular el período, la frecuencia, los tiempos de subida y bajada y los tiempos de retardo en la propagación (subida y bajada) de una puerta NAND con una entrada conectada a un "1" y la otra conectada a una señal cuyas características junto con las de salida se representan en la figura 6.17 (Tiempos en nanosegundos).*

Figura 6.16

Solución:

Período: $T=95+120=215^{ns}$

Frecuencia: $F=1/T=1/215^{ns}=1/0,000000215^{s}=4651162,79^{Hz}=4,65^{MHz}$

Tiempo de subida: $t_s=7^{ns}$

Tiempo de bajada: $t_b=9^{ns}$

Tiempo de retardo en la propagación:

$$t_{PLH}=24^{ns}$$

$$t_{PHL}=22^{ns}$$

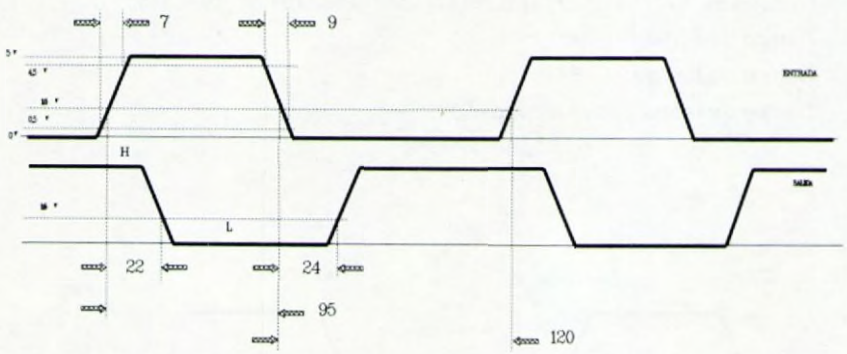


Figura 6.17

TEMA 7
CIRCUITOS COMBINACIONALES

7

CIRCUITOS COMBINACIONALES

7.1. INTRODUCCION

Después de analizar los sistemas de numeración, la lógica algebraica y la formación de funciones, realizando algunos circuitos de iniciación con puertas lógicas, se hace necesario introducir un nivel mayor de complejidad, definiendo bloques genéricos constituídos por circuitos integrados. Estos bloques están formados por los elementos denominados codificadores, decodificadores, multiplexores, demultiplexores, comparadores, etc. y sus variantes. Su constitución interna, como se verá más adelante, está formada simplemente por un conjunto interconexionado de puertas lógicas.

Debido a la gran difusión y utilización de este tipo de aplicaciones integradas, su variedad es enorme y sus posibilidades muy amplias, al reducirse con ellos enormemente el número de conexiones a realizar y el número de circuitos integrados y puertas lógicas que se deben utilizar. A lo largo del tema se definirán los conceptos básicos que determinan dichos circuitos, sus tablas de funcionamiento y sus aplicaciones principales, indicando gráficamente mediante esquemas los conexionados que se deben realizar.

7.2. FORMACION DE UN CIRCUITO COMBINACIONAL

Un circuito combinacional es básicamente un conjunto de puertas lógicas conexionadas de tal manera que los valores que se obtienen en las salidas del mismo dependen únicamente de los valores que en cada instante tengan las entradas. No están influenciados por la evolución temporal, lo que quiere decir que si no cambian las entradas no cambiarán las salidas.

Las funciones lógicas o algebraicas, definidas así por la utilización de unas variables de entrada que se basan en la lógica booleana permiten que mediante operaciones con los términos que las definen se puede acceder a

la construcción de un circuito con puertas lógicas elementales. Ahora, estas puertas lógicas se agrupan definiendo unos conjuntos o bloques lógicos con denominación propia. Estos bloques seguirán dependiendo de las variables de entrada que definían una función, ya que nada se modifica al realizar la mencionada agrupación.

7.3. **DECODIFICADORES**

El decodificador es un circuito combinacional formado por **n** entradas y 2^n salidas, y funciona de tal forma que **PARA CADA COMBINACION BINARIA DE LAS ENTRADAS SOLAMENTE UNA SALIDA ESTARA ACTIVADA**.

Para 1 entrada existen 2 salidas, para 2 entradas 4 salidas, para 3 entradas 8 salidas, etc. Esto quiere decir que el número de salidas se corresponde con el número posible de combinaciones de las variables de entrada, bien en forma algebráica o numérica. Con los datos indicados, se pueden construir las tablas de verdad, de donde se obtienen las ecuaciones que permiten el diseño y construcción de los circuitos correspondientes.

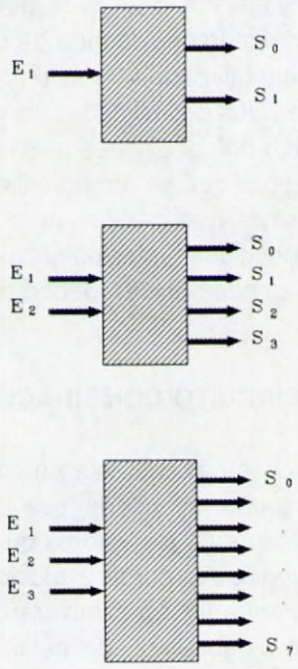


Figura 7.1. Bloques decodificadores

E_2	E_1	S_3	S_2	S_1	S_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Tabla 7.1. Decodificador 2x4

$$S_0 = \bar{E}_2 \bar{E}_1 \qquad [7.1]$$

$$S_1 = \bar{E}_2 E_1 \qquad [7.2]$$

$$S_2 = E_2 \bar{E}_1 \qquad [7.3]$$

$$S_3 = E_2 E_1 \qquad [7.4]$$

La representación de las ecuaciones mediante puertas lógicas conduce al circuito representado en la figura 7.2.

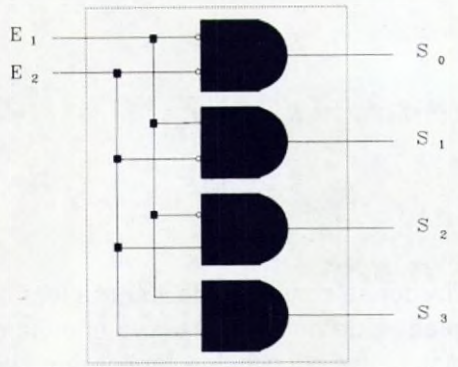


Figura 7.2. Decodificador 2x4

Si se aplica el mismo procedimiento a un número superior de variables de entrada y salida, por ejemplo, 3 entradas y 8 salidas, se obtienen los resultados de la tabla 7.2.

E_3	E_2	E_1	S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Tabla 7.2. Decodificador 3x8

$$S_0=\bar{E}_3\bar{E}_2\bar{E}_1 \quad [7.5]$$

$$S_1=\bar{E}_3\bar{E}_2E_1 \quad [7.6]$$

$$S_2=\bar{E}_3E_2\bar{E}_1 \quad [7.7]$$

$$S_3=\bar{E}_3E_2E_1 \quad [7.8]$$

$$S_4=E_3\bar{E}_2\bar{E}_1 \quad [7.9]$$

$$S_5=E_3\bar{E}_2E_1 \quad [7.10]$$

$$S_6=E_3E_2\bar{E}_1 \quad [7.11]$$

$$S_7=E_3E_2E_1 \quad [7.12]$$

Cualquier decodificador se construye de idéntica forma a los diseños realizados para 2 y 3 variables de entrada. Dada la enorme cantidad de puertas lógicas que se emplean, estos circuitos se simbolizan mediante bloques donde se indican las entradas y salidas de los mismos, como puede apreciarse en la figura 7.1. Es importante a la hora de analizar un diagrama de bloques, el orden de las combinaciones de entrada, que puede dar lugar a distintas combinaciones de salidas, por lo que la tabla de verdad es la que determina el funcionamiento exacto del circuito.

En resumen, se puede indicar que un decodificador es un **CONVERSOR DE BINARIO A DECIMAL**, dado que la salida que se obtiene es la expresión

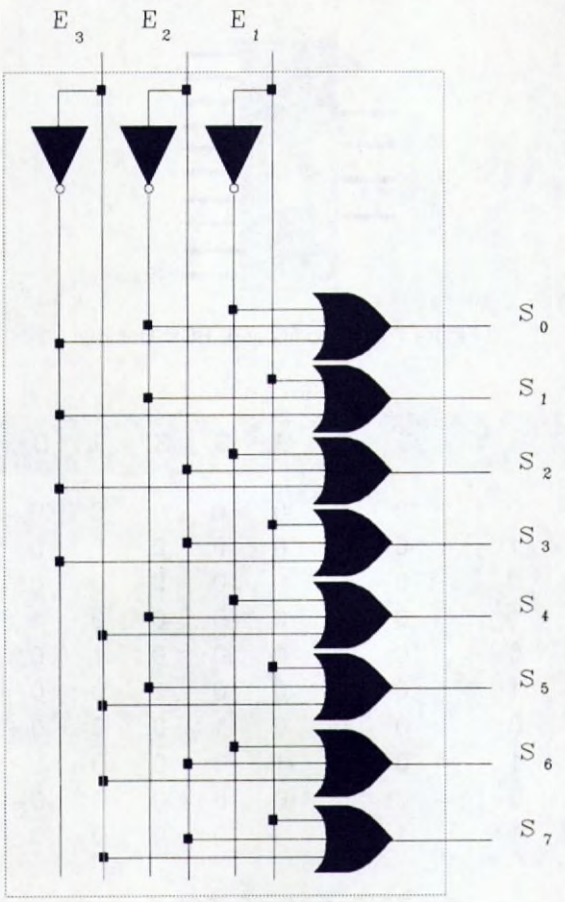


Figura 7.3. Decodificador 3x8

decimal de la combinación binaria que forman las variables de entrada. Basado en este principio, se puede determinar un decodificador particular, que es el más utilizado de todos, y que se corresponde con la conversión de 4 dígitos binarios a decimal, pero utilizando sólo 10 salidas, correspondientes a los 10 elementos constitutivos del sistema decimal, del 0 al 9.

Como las agrupaciones de 4 dígitos binarios se corresponden con el sistema de numeración BCD, este decodificador se conoce normalmente como **DECODIFICADOR BCD-DECIMAL** o **CONVERTIDOR DE CODIGO BCD A DECIMAL**. Su tabla de funcionamiento es similar a la del decodificador de 4 entradas y 16 salidas, teniendo en cuenta que sólo se utilizan las salidas S_0 a S_9 y que las salidas S_{10} a S_{15} no existen, siendo el único decodificador que no cumple la relación "n" entradas/"2^n" salidas. En la figura 7.4 se representa el esquema bloque correspondiente a dicho decodificador.

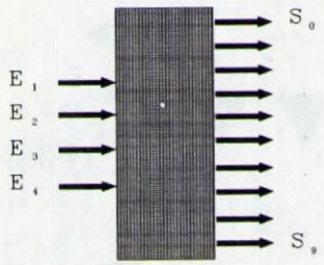


Figura 7.4. Decodificador BCD-Decimal

E_4	E_3	E_2	E_1	S_9	S_8	S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0
0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	0	1	0	0	0	0	0	0
0	1	1	1	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	0	0	0	0

Tabla 7.3. Decodificador BCD-Decimal

En caso de tener que decodificar un número elevado de salidas, se recurre a la agrupación de decodificadores simples como los que se han visto, dado que la fabricación de circuitos integrados de mayor número de conexiones externas no se suele realizar con fines comerciales. Los circuitos integrados decodificadores más empleados son los de 3x8 y 4x10 (BCD-Decimal). Para realizar montajes múltiples se suelen emplear decodificadores BCD/Decimal, de los que sólo se emplean sus 8 primeras salidas, S_0 a S_7. En todos ellos, las tres entradas menos significativas se conectan en paralelo, utilizándose la cuarta entrada para seleccionar el decodificador que proporcionará la única salida activa. Esta selección se puede realizar mediante otros decodificadores.

En las figuras 7.5 y 7.6 se representan los circuitos correspondientes a 16 y 32 salidas decodificadas. Una de las aplicaciones de los decodificadores es la iluminación de las pantallas de calculadoras, marcadores luminosos, conta-

dores digitales, displays de todo tipo etc., donde se precisa obtener la visualización numérica de un determinado dato.

7.3.1. **Generación de funciones con decodificadores**

Si cada salida del decodificador corresponde a un valor decimal comenzando por el valor cero, cada una de ellas puede hacerse equivalente a un término canónico de suma de productos de los que definen una función lógica. Si la expresión del número de entradas y de salidas de un decodificador

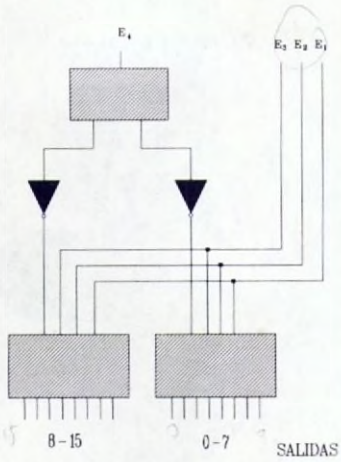


Figura 7.5. Montaje para decodificación 4x16

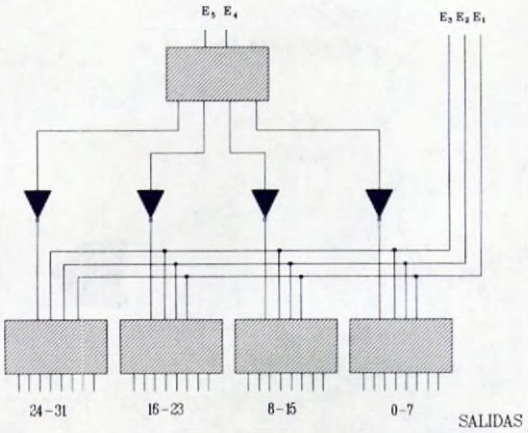


Figura 7.6. Montaje para decodificación 5x32

coincide con el número de variables y de términos definidos por las mismas, quiere decir que dicho circuito puede ser utilizado como generador de funciones lógicas y por tanto sólo es necesario relacionar entre sí los conceptos función lógica y decodificador para obtener la relación de diseño del circuito final. Al tratarse de la primera forma canónica, sus términos necesitan ser sumados, por lo que se conectarán los terminales de salida del decodificador a una puerta OR.

*## **Ejemplo 7.1**: Generar con un decodificador de 3 entradas la siguiente función lógica:*

$$F(A,B,C)=m_0+m_2+m_5+m_7$$

Solución:

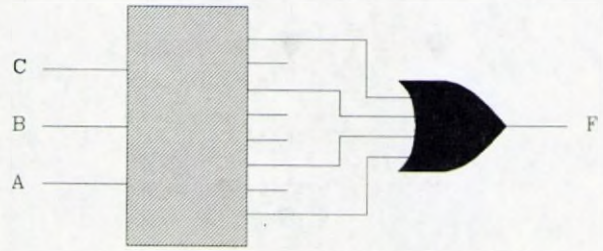


Figura 7.7.

*## **Ejemplo 7.2**: Generar con decodificadores la siguiente función lógica:*

$$F(A,B,C)=M_0M_1M_2M_3M_6$$

Solución:

$$F(A,B,C)=m_0+m_2+m_3$$

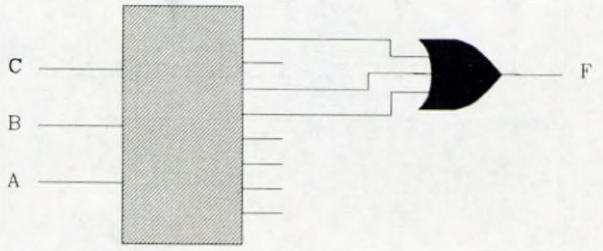


Figura 7.8

Ejemplo 7.3: *Construir un decodificador de 6x64 con decodificadores BCD-Decimal.*

Solución:

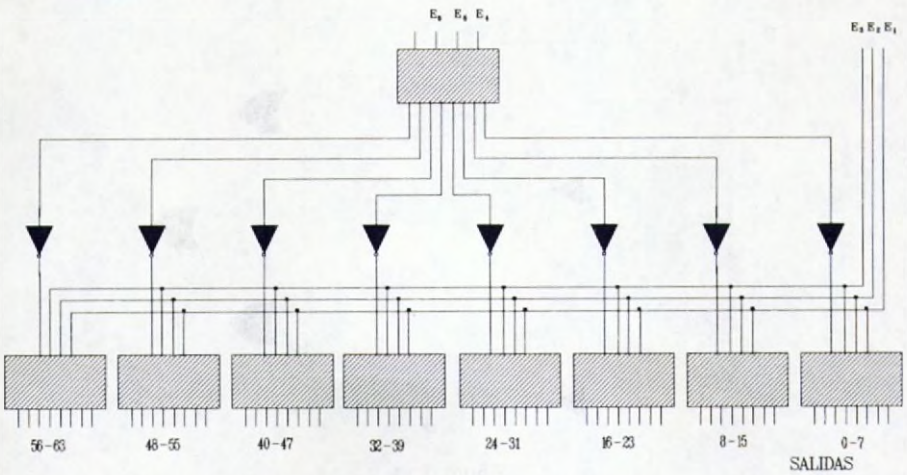


Figura 7.9

Ejemplo 7.4: *Construir el circuito de puertas lógicas del decodificador BCD-Decimal.*

Solución:

De la tabla 7.3, aplicando funciones incompletamente especificadas, se obtienen las siguientes ecuaciones, que conducen al circuito de la figura 7.10:

$$S_0=\bar{E}_4\bar{E}_3\bar{E}_2\bar{E}_1$$

$$S_1=\bar{E}_4\bar{E}_3\bar{E}_2E_1$$

$$S_2=\bar{E}_3E_2\bar{E}_1$$

$$S_3=\bar{E}_3E_2E_1$$

$$S_4=E_3\bar{E}_2\bar{E}_1$$

$$S_5=E_3\bar{E}_2E_1$$

$$S_6=E_3E_2\bar{E}_1$$

$$S_7=E_3E_2E_1$$

$$S_8=E_3\bar{E}_1$$

$$S_9=E_3E_1$$

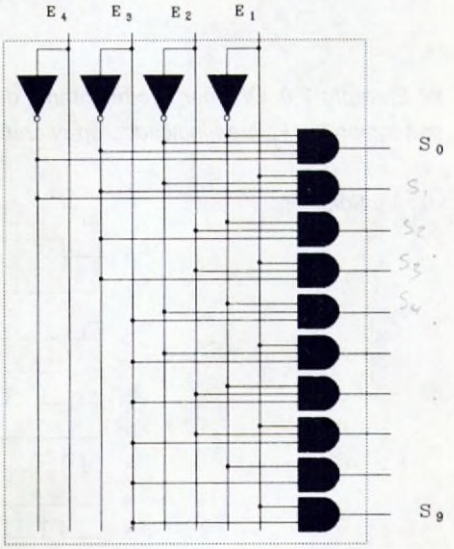


Figura 7.10

*## **Ejemplo 7.5**: Obtener la función algebraica más simple del circuito representado en la figura 7.11.*

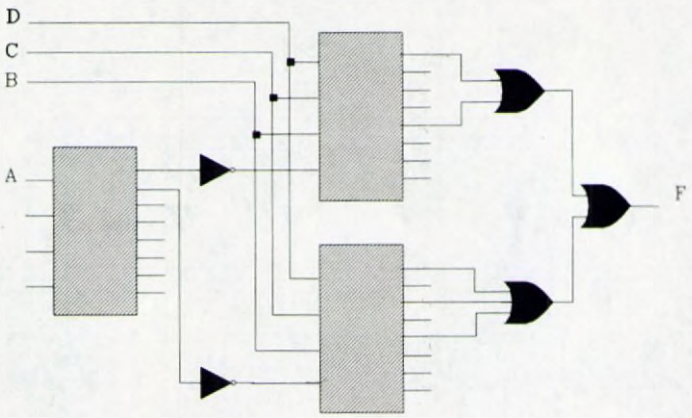


Figura 7.11

Solución:

$$F(A,B,C,D)=m_0+m_4+m_8+m_{10}+m_{12}$$

Simplificando por Karnaugh se obtiene la siguiente ecuación:

$$F(A,B,C,D)=\bar{C}\bar{D}+A\bar{B}\bar{D}$$

*## **Ejemplo 7.6**: Obtener el cronograma de salida del decodificador 3x8 cuando sus entradas toman los valores indicados en el cronograma de la figura 7.12.*

Solución:

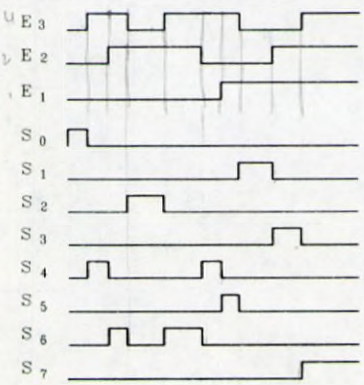


Figura 7.12

***## Ejemplo 7.7**: Obtener el cronograma de salida del decodificador múltiple 4x16 de la figura 7.5 cuando sus entradas toman los valores del cronograma de la figura 7.13.*

Solución:

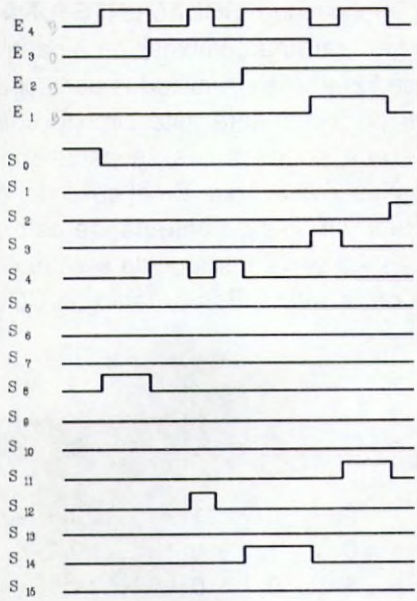


Figura 7.13

***## Ejemplo 7.8**: Aplicar al problema 7.1 el cronograma de la figura 7.14 y obtener las salidas correspondientes.*

Solución:

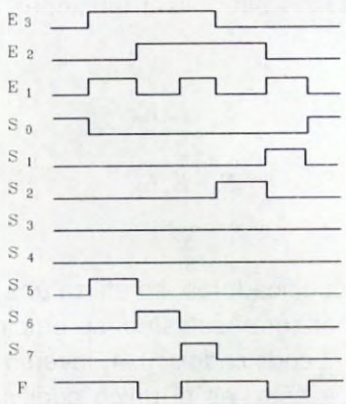


Figura 7.14

7.4. CODIFICADORES

El circuito combinacional definido como codificador está constituído por 2^n entradas y **n** salidas. Por tanto, se corresponde con el circuito inverso del decodificador, lo que implica que **SOLAMENTE UNA ENTRADA PODRA SER ACTIVADA** para generar una combinación binaria de salida.

Tomando como base los análisis realizados para el decodificador, se puede entender y analizar perfectamente este circuito combinacional, teniendo en cuenta únicamente que donde antes se hacía mención a una entrada ahora se hace a una salida y viceversa. En el caso del codificador, al introducir una señal en cualquier variable de entrada, se obtendrá una combinación de valores en las salidas. La señal introducida será el equivalente decimal de la combinación binaria de la salida. Es por tanto un **CONVERSOR DECIMAL A BINARIO**.

E_3	E_2	E_1	E_0	S_2	S_1
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Tabla 7.4. Codificador 4x2

Aplicando funciones incompletamente especificadas para simplificar la tabla 7.4, se obtienen las ecuaciones [7.13] y [7.14] que conducen a la figura 7.15.(b), un circuito bastante más simple que el 7.15.(a) obtenido con los valores "1" de la tabla 7.4 sin aplicar funciones incompletamente especificadas:

$$S_1 = \bar{E}_2 \bar{E}_0 \qquad [7.13]$$

$$S_2 = \bar{E}_1 \bar{E}_0 \qquad [7.14]$$

Al igual que en los decodificadores, se utiliza una representación simbólica mediante bloques genéricos. Asimismo, se emplea la denominación Decimal-BCD para calificar al codificador 10x4, inverso del decodificador BCD-Decimal, que al igual que éste, es el único codificador que no cumple la relación "2^n" entradas/"n" salidas. Los diagramas de bloques correspondien-

tes a 2, 4 y 8 entradas son los indicados en la figura 7.16, mientras que el Decimal/BCD se representa en la figura 7.17.

Si se emplean más de 10 entradas, se deberá utilizar el conexionado múltiple de codificadores. En la figura 7.19 se representa como ejemplo el codificador de 16 entradas, empleando codificadores 8x3, de manera equivalente a los montajes decodificados. Este tipo de circuito tiene una cuarta salida de "Activación o Prioridad", que proporciona un "1" cuando alguna de sus entradas es activada, indicando de esta forma el codificador que está actuando y determinando la salida S_3 del montaje múltiple.

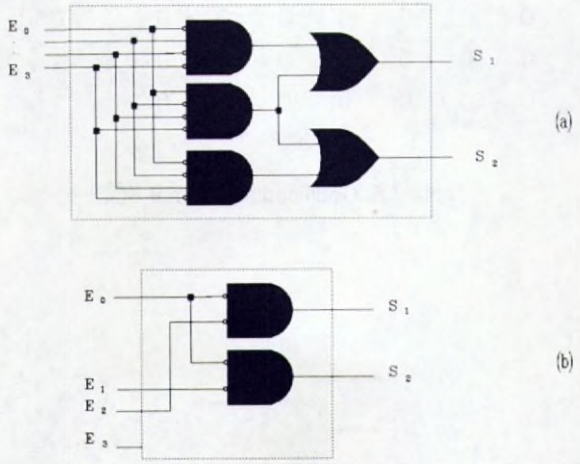


Figura 7.15. Codificador 4x2

E_7	E_6	E_5	E_4	E_3	E_2	E_1	E_0	S_3	S_2	S_1
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Tabla 7.5. Codificador 8x3

E_9	E_8	E_7	E_6	E_5	E_4	E_3	E_2	E_1	E_0	S_4	S_3	S_2	S_1
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

Tabla 7.6. Codificador Decimal-BCD

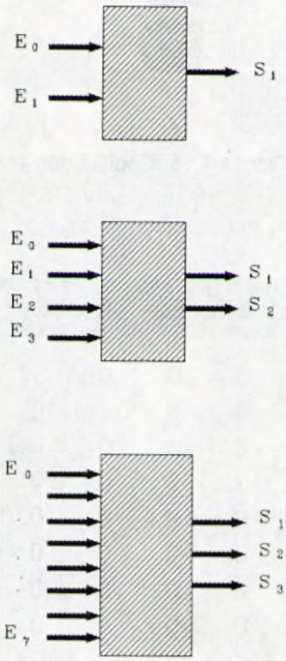


Figura 7.16. Diagramas bloque de codificadores

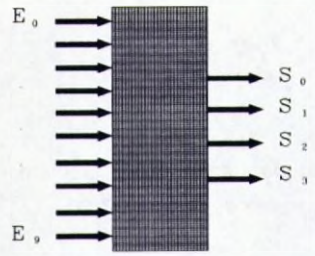


Figura 7.17. Codificador Decimal/BCD

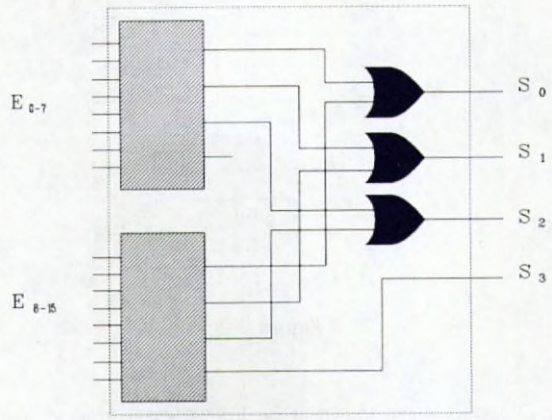


Figura 7.18. Montaje múltiple para codificación 16x4

***## Ejemplo 7.9**: Analizar las salidas de cada codificador elemental de la figura 7.19 si se introduce un "1" en x.*

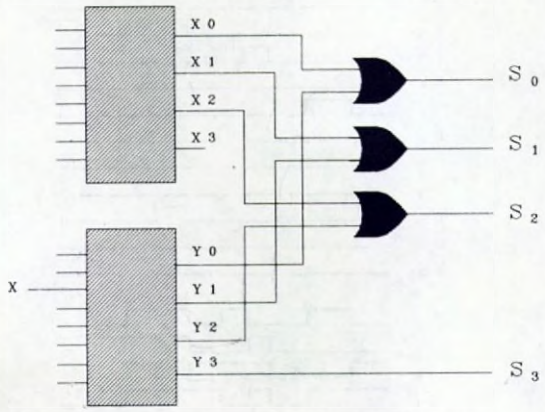


Figura 7.19

Solución:

$X_0=0$ $X_1=0$ $X_2=0$ $X_3=0$

$Y_0=0$ $Y_1=1$ $Y_2=0$ $Y_3=1$

*## **Ejemplo 7.10**: Obtener el cronograma de salida del codificador 8x3 cuando sus entradas toman los valores del cronograma de la figura 7.20.*

Solución:

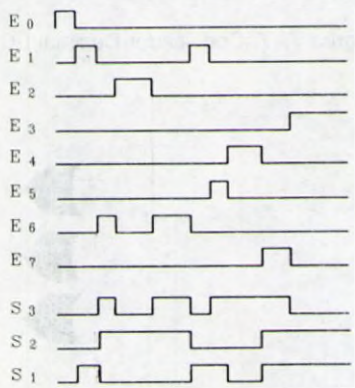


Figura 7.20

*## **Ejemplo 7.11**: Obtener el cronograma de las salidas del codificador múltiple 16x4 al aplicar las entradas del cronograma de la figura 7.21.*

Solución:

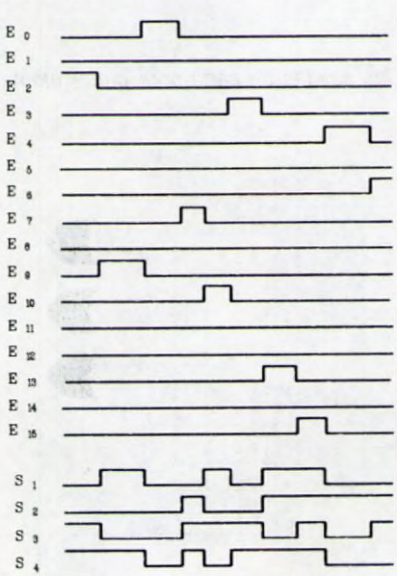


Figura 7.21

7.5. **MULTIPLEXORES**

El multiplexor es un dispositivo constituido como circuito combinacional, que dispone de **n** entradas de selección, **2^n** entradas de datos y **1** salida. Las entradas de selección permiten que **UNA SOLA DE LAS VARIABLES DE ENTRADA SE TRANSMITA A LA SALIDA** con cada combinación binaria.

Cada variable de entrada se corresponde con una combinación de los selectores según se indica en las tablas 7.7, donde se representan las variables de entrada de datos como "E", la salida como "S" y las entradas de selección o selectores como "C". La tabla 7.7.(a) coresponde a la tabla completa y la 7.7.(c) a la tabla resumida, donde se observa que para C_1=0, se transmite a la salida el valor de E_0, y para C_1=1 se transmite el valor de E_1 ("0" ó "1"). La tabla 7.7.(a) conduce a la obtención de la ecuación de diseño del circuito combinacional representado en la figura 7.22.

E_1	E_0	C_1	S
0	0	0	0
0	1	0	1
1	0	0	0
1	1	0	1
0	0	1	0
0	1	1	0
1	0	1	1
1	1	1	1

Tabla 7.7.(a). Multiplexor de 2 canales

E_1	E_0	C_1	S
-	0	0	0
-	1	0	1
0	-	1	0
1	-	1	1

Tabla 7.7.(b). Multiplexor de 2 canales

E_1	E_0	C_1	S
-	E_0	0	E_0
E_1	-	1	E_1

Tabla 7.7.(c). Multiplexor de 2 canales

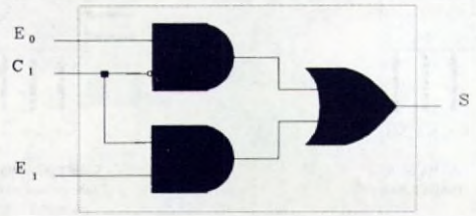


Figura 7.22. Multiplexor de 2 canales

Utilizando simplificación algebraica en la tabla 7.7.(a), se obtiene la siguiente ecuación para el multiplexor de 2 canales:

$$S=E_0\bar{C}_1+E_1C_1 \qquad [7.15]$$

El funcionamiento de este tipo de circuito combinacional es similar a la de un conmutador que transmite a la salida una sola de las señales de entrada, dependiendo únicamente de la posición de conexión de dicho conmutador. Este posicionamiento es el equivalente a la selección mediante las combinaciones binarias de las entradas selectoras. El esquema representativo de dichos dispositivos es el indicado en la figura 7.23, donde se dibujan los esquemas bloques de 2, 4 y 8 variables de entrada.

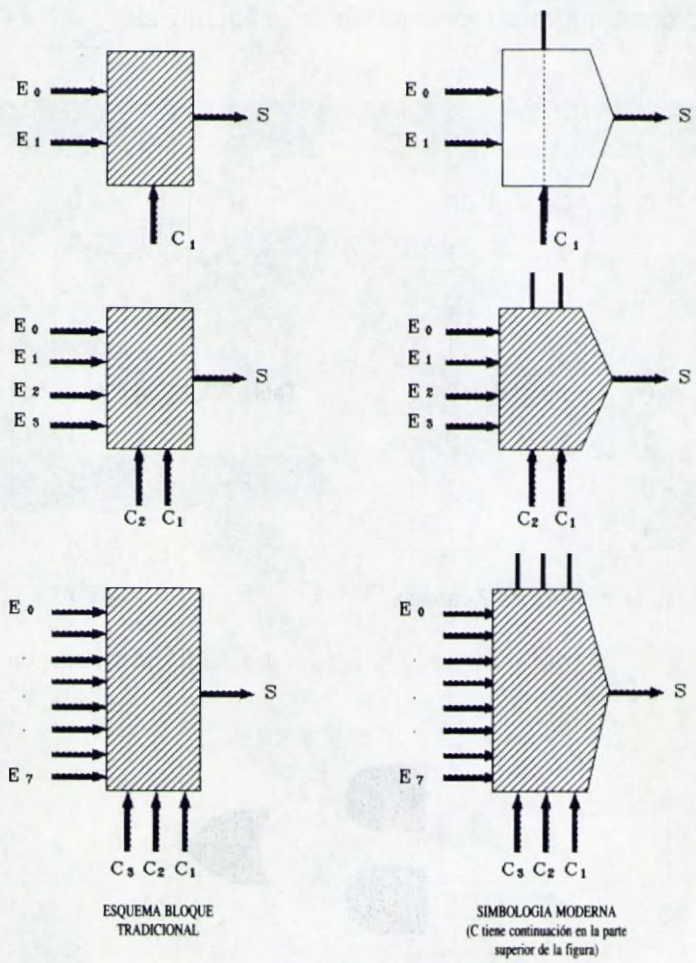


Figura 7.23. Bloques multiplexores

E_3	E_2	E_1	E_0	C_2	C_1	S
-	-	-	E_0	0	0	E_0
-	-	E_1	-	0	1	E_1
-	E_2	-	-	1	0	E_2
E_3	-	-	-	1	1	E_3

Tabla 7.8. Multiplexor de 4 canales

E_7	E_6	E_5	E_4	E_3	E_2	E_1	E_0	C_3	C_2	C_1	S
-	-	-	-	-	-	-	E_0	0	0	0	E_0
-	-	-	-	-	-	E_1	-	0	0	1	E_1
-	-	-	-	-	E_2	-	-	0	1	0	E_2
-	-	-	-	E_3	-	-	-	0	1	1	E_3
-	-	-	E_4	-	-	-	-	1	0	0	E_4
-	-	E_5	-	-	-	-	-	1	0	1	E_5
-	E_6	-	-	-	-	-	-	1	1	0	E_6
E_7	-	-	-	-	-	-	-	1	1	1	E_7

Tabla 7.9. Multiplexor de 8 canales

En el lenguaje normal de los dispositivos electrónicos digitales, se denominan **CANALES** a las variables de entrada de datos de un multiplexor, con lo cuál, los diagramas de bloques representados en la figura 7.26 se denominarían de 2, 4 y 8 canales respectivamente. En caso de utilizar más de 8 canales o entradas, se necesita recurrir a un montaje múltiple, similar al efectuado para los decodificadores y codificadores, utilizando varios multiplexores de 2, 4 u 8 canales. En las figuras 7.24 y 7.25 se representan los esquemas de conexionado para multiplexar 16 y 32 canales empleando diferentes circuitos básicos. Usualmente se abrevia la expresión multiplexor utilizando la denominación MUX.

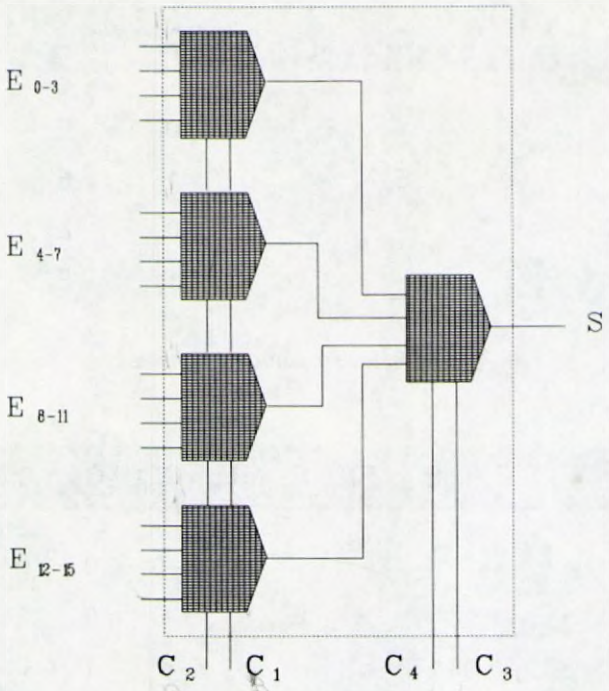


Figura 7.24. Montaje de multiplexación de 16 canales

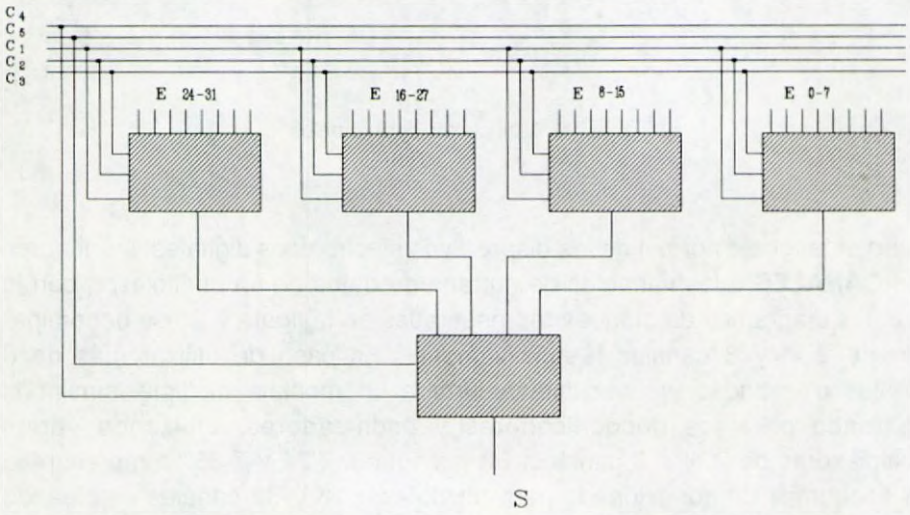


Figura 7.25. Montaje de multiplexación de 32 canales

7.5.1. **Generación de funciones con multiplexores**

Al igual que en caso del decodificador, existe relación entre los datos manipulados por el multiplexor y los términos que definen una función lógica. Los terminales que equivalen a los términos de la función son tanto las entradas de datos como las entradas de selección. La utilización de ambas entradas posibilita la generación de funciones de mayor número de términos si se comparan con circuitos decodificadores equivalentes. La asignación de variables definitorias de una función con las entradas del multiplexor puede hacerse de la siguiente forma:

* Una variable se asigna a las entradas de datos

* Las otras variables se asignan a las entradas de selección

* Se constituye una tabla con 2 filas horizontales, correspondientes a la variable asignada a las entradas de datos y tantas columnas como combinaciones binarias de las variables asignadas a las entradas de selección.

* Cada columna de dicha tabla se hace equivalente a una entrada de datos del multiplexor, colocadas en orden creciente de izquierda a derecha. Puede alterarse el orden de las columnas utilizando los valores de Karnaugh que varían únicamente en un bit, pero hay que tener en cuenta que el orden de las entradas E_0,..., E_7 asignado a dicha tabla ha de ser modificado.

* Se rellena la tabla con los valores correspondientes a la función lógica.

* Se analizan los datos de cada columna de la tabla para obtener los términos E_i, pudiendo obtenerse "0" (ambas celdas son "0"), "1" (ambas celdas son "1"), A (hay un "1" en la fila correspondiente a A=1) y complemento de A (hay un "1" en la fila correspondiente a A=0).

* Se conectan las entradas de datos y de selección de acuerdo con los valores obtenidos en los diferentes E_i y las variables asignadas a los selectores.

Los esquemas para 3 y 4 variables, incluyendo el valor decimal de cada celda son los siguientes:

A \ B C	00	01	10	11
0	0	1	2	3
1	4	5	6	7
	E_0	E_1	E_2	E_3

Tabla 7.10. Función de 3 variables

A \ B C D	000	001	010	011	100	101	110	111
0	0	1	2	3	4	5	6	7
1	8	9	10	11	12	13	14	15
	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7

Tabla 7.11. Función de 4 variables

***## Ejemplo 7.12**: Diseñar con multiplexores el circuito que genere la siguiente función lógica de 3 variables:*

$$F(A,B,C)=m_0+m_3+m_5+m_7$$

Solución:

A \ B C	00	01	10	11
0	1	0	0	1
1	0	1	0	1
	E_0	E_1	E_2	E_3

$E_0=\bar{A}$ $E_1=A$ $E_2=0$ $E_3=1$

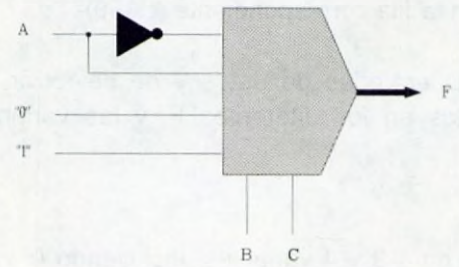


Figura 7.26

*## **Ejemplo 7.13**: Diseñar con multiplexores el circuito que genere la siguiente función lógica de 4 variables:*

$$F(A,B,C,D)=m_0+m_3+m_6+m_8+m_{10}+m_{13}+m_{15}$$

Solución:

A \ B C D	000	001	010	011	100	101	110	111
0	1	0	0	1	0	0	1	0
1	1	0	1	0	0	1	0	1
	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7

$E_0=1 \quad E_1=0 \quad E_2=A \quad E_3=\overline{A}$
$E_4=0 \quad E_5=A \quad E_6=\overline{A} \quad E_7=A$

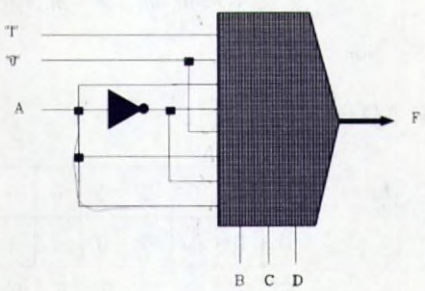


Figura 7.27

*## **Ejemplo 7.14**: Construir con multiplexores de 2 canales la función lógica siguiente:*

$$F(A,B,C)=M_0M_2M_6M_7$$

Solución:

A \ B C	00	01	10	11
0	0	0	1	1
1	1	0	1	0
	E_0	E_1	E_2	E_3

$E_0=A \quad E_1=0 \quad E_2=1 \quad E_3=\overline{A}$

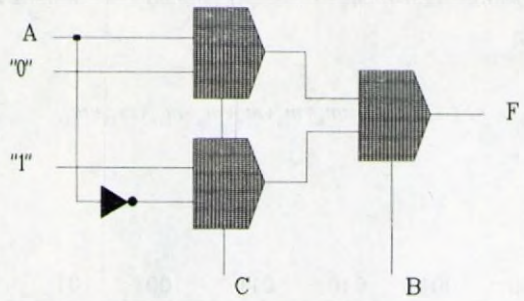


Figura 7.28

*## **Ejemplo 7.15**: Construir con multiplexores la siguiente función lógica de 5 variables:*

$$F=m_1+m_7+m_{14}+m_{15}+m_{18}+m_{22}+m_{23}+m_{24}+m_{27}+m_{31}$$

Solución:

A \ B C D E	0000															1111
0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	1
1	0	0	1	0	0	0	1	1	1	0	0	1	0	0	0	1
	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}	E_{11}	E_{12}	E_{13}	E_{14}	E_{15}

$$E_0=0 \quad E_1=\overline{A} \quad E_2=A \quad E_3=0 \quad E_4=0 \quad E_5=0 \quad E_6=A \quad E_7=1$$
$$E_8=A \quad E_9=0 \quad E_{10}=0 \quad E_{11}=A \quad E_{12}=0 \quad E_{13}=0 \quad E_{14}=\overline{A} \quad E_{15}=1$$

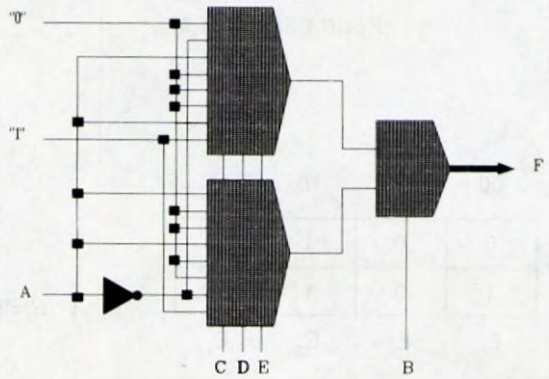


Figura 7.29

*## **Ejemplo 7.16**: Construir un multiplexor de 8 canales empleando multiplexores de 2 canales.*

*## **Ejemplo 7.17**: Aplicar el cronograma de la figura 7.31 al problema 7.14, obteniendo el cronograma de salida.*

*## **Ejemplo 7.18**: Aplicar el cronograma de la figura 7.32 a la figura 7.24, obteniendo su salida.*

Soluciones:

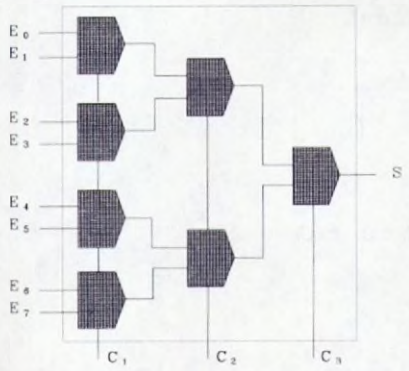


Figura 7.30

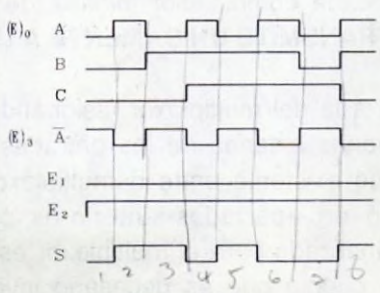


Figura 7.31

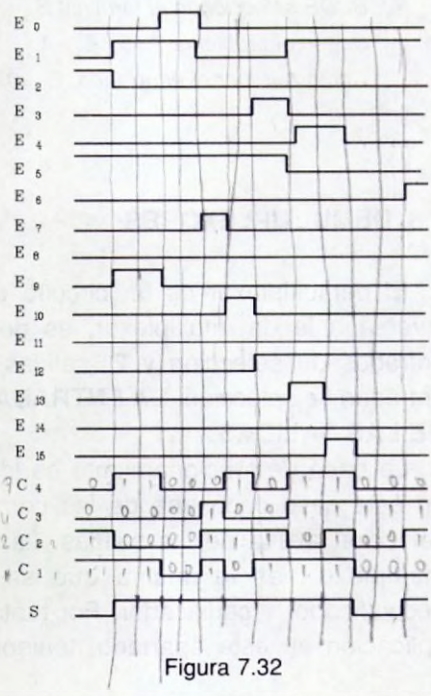


Figura 7.32

*## **Ejemplo 7.19**: Aplicar el cronograma de la figura 7.33 al circuito del ejemplo 7.15 y obtener su salida.*

Solución:

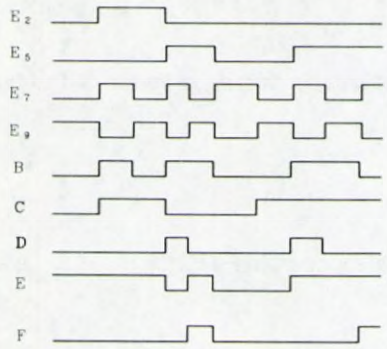


Figura 7.33

*## **Ejemplo 7.20**: Obtener la salida del circuito del ejemplo 7.15 cuando las variables toman los siguientes valores: A=0, B=1, C=0, D=1, E=1.*

Solución:

BCDE seleccionan el terminal E_{11}
Según el problema 7.15, E_{11}=A
Como A=0 por el enunciado, E_{11}=0, y por tanto, F=0

7.6. **DEMULTIPLEXORES**

El demultiplexor es un circuito combinacional que efectua la operación inversa a la del multiplexor, es decir, dispone de **1** entrada de datos, **n** entradas de selección y 2^n salidas. Con cada combinación binaria de las entradas de selección, **LA ENTRADA SE TRANSMITE UNICAMENTE A UNA DE LAS SALIDAS**.

La base de funcionamiento es idéntica a la del multiplexor, asignándose en este caso cada una de las combinaciones binarias de las entradas de selección a una de las salidas. La relación existente entre demultiplexor y multiplexor, es la misma que se indicó en apartados anteriores para decodificador y codificador. Por tanto, lo analizado para el multiplexor es de aplicación en este apartado, teniendo en cuenta que es necesario invertir

entradas y salidas para realizar el funcionamiento y análisis correcto. Las tablas de funcionamiento del demultiplexores de 2 salidas son la 7.12.(a) y la 7.12.(b), correspondientes a las tablas completa y resumida (se representa para valores de "E" iguales a "0" y "1").

E	C_1	S_1	S_0
0	0	0	0
1	0	0	1
0	1	0	0
1	1	1	0

Tabla 7.12.(a). Demultiplexor de 2 canales

E	C_1	S_1	S_0
E	0	0	E
E	1	E	0

Tabla 7.12.(b). Demultiplexor de 2 canales

$$S_0 = E\overline{C_1} \quad [7.16]$$

$$S_1 = EC_1 \quad [7.17]$$

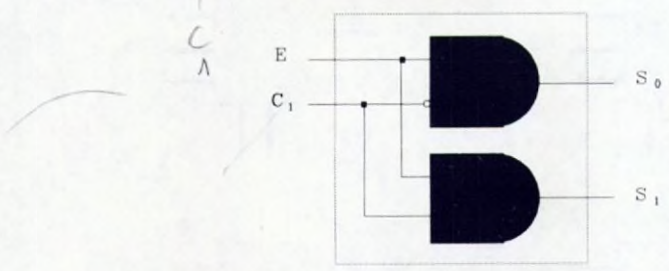


Figura 7.34. Demultiplexor de 4 canales

E	C_2	C_1	S_3	S_2	S_1	S_0
E	0	0	0	0	0	E
E	0	1	0	0	E	0
E	1	0	0	E	0	0
E	1	1	E	0	0	0

Tabla 7.13. Demultiplexor de 4 canales

Al igual que en los apartados anteriores, este tipo de circuito combinacional se representa también mediante diagramas de bloques, con indicación de entradas de datos, salidas y entradas selectoras. La denominación del término canal en demultiplexación se aplica a las salidas de los circuitos. En la figura 7.35 se representan los diagramas correspondientes a 2, 4 y 8 salidas. Los montajes que posean un número elevado de salidas, necesitarán un conexionado similar al realizado para los multiplexores, combinando entre sí varios bloques demultiplexores de 2, 4 u 8 salidas.

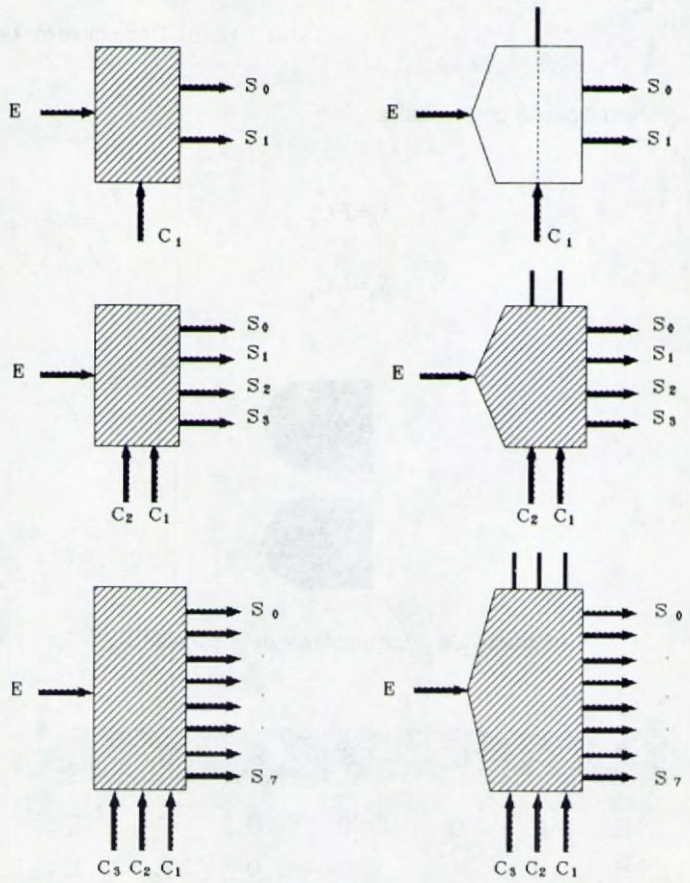


Figura 7.35

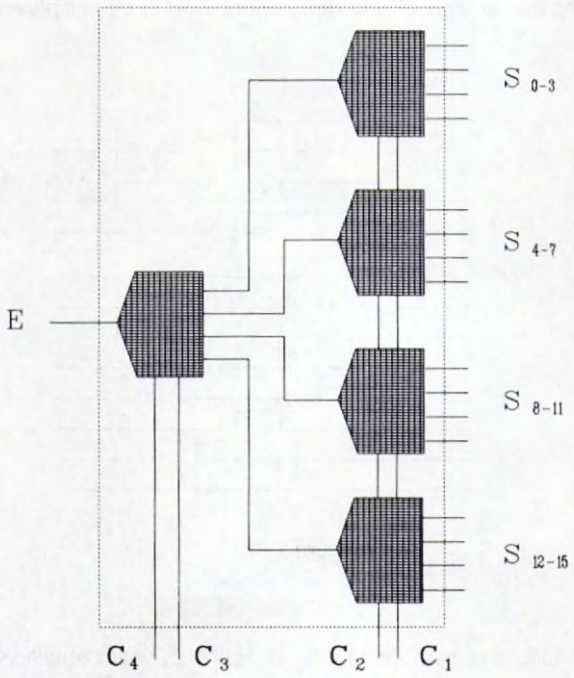


Figura 7.36. Montaje múltiple de 16 canales demultiplexados

***## Ejemplo 7.21**: Construir el esquema de un demultiplexor de 8 canales con demultiplexores de 2 canales.*

Solución:

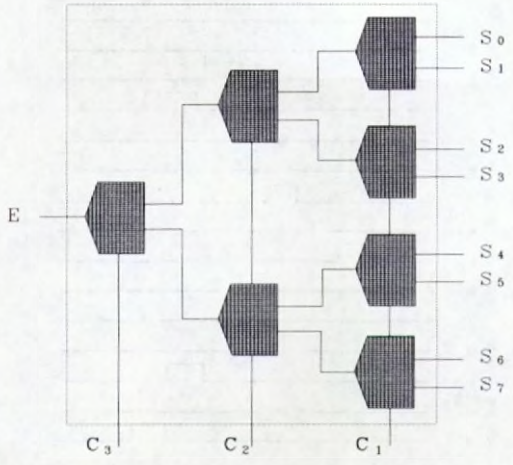


Figura 7.37

***## Ejemplo 7.22**: Aplicar el cronograma de la figura 7.38 al demultiplexor de 8 canales y obtener su salida.*

Solución:

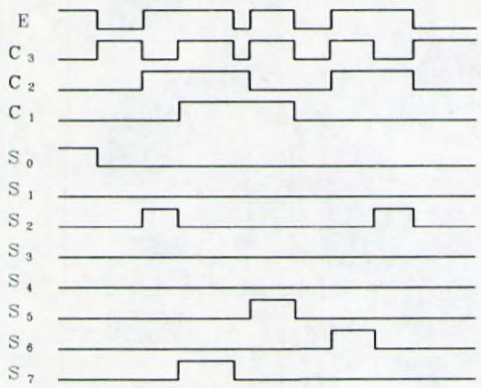


Figura 7.38

***## Ejemplo 7.23**: Aplicar el cronograma de la figura 7.39 al demultiplexor múltiple del ejemplo 7.21 y obtener su salida.*

Solución:

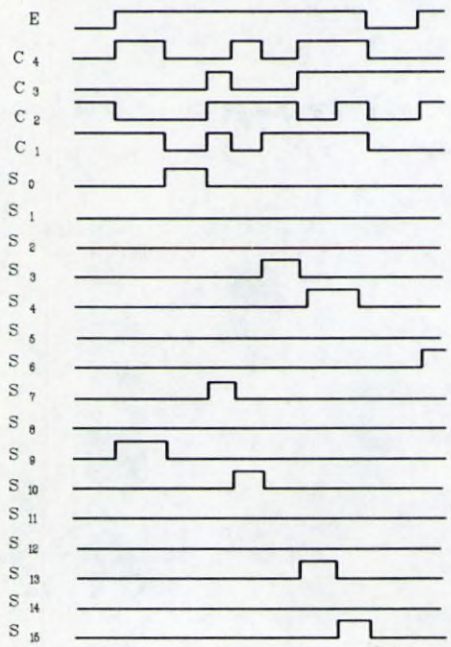


Figura 7.39

7.7. CONVERTIDORES DE CODIGO

Un circuito combinacional bastante interesante desde el punto de vista de su aplicación, es el convertidor de código, circuito derivado del decodificador, que transforma una señal de entrada en un sistema de numeración o código particular a otra de diferente sistema o código. Existen numerosos convertidores de código, algunos de los cuáles se designan con otras denominaciones, como el decodificador o convertidor de sistema binario a decimal y el codificador o convertidor de sistema decimal a binario.

El más representativo de los convertidores de código es el que se aplica a la iluminación del display 7 segmentos como el representado en la figura 7.40, teniendo en cuenta que cada segmento está referenciado por una letra minúscula. El convertidor de código de esta aplicación necesita transformar los valores binarios de la entrada a 7 salidas decodificadas, que serán las que conectarán con el display luminoso.

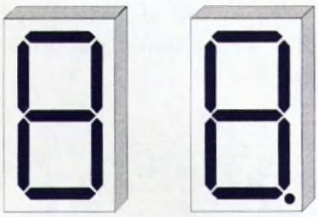


Figura 7.40. Displays de 7 y 8 segmentos

Teniendo en cuenta que se necesitan 4 códigos binarios para representar las combinaciones que iluminan el display, el convertidor de código tendrá 4 entradas y 7 salidas. Por tanto, la tabla de funcionamiento del mismo es la 7.15, relacionando las entradas binarias del convertidor con sus salidas y con las entradas del display 7 segmentos.

Una vez definidos el display y el convertidor de código, la conexión es inmediata uniendo las salidas del convertidor con las entradas del display luminoso, tal como se observa en la figura 7.41. Esta conexión, dependiendo de los circuitos, suele necesitar una resistencia limitadora conectada entre las salidas del convertidor de código y las entradas del display, para evitara que este último reciba un exceso de corriente que pueda dañarlo.

Cuando se habla de display 7 segmentos, se interpreta que dicha estructura es de aplicación no sólo a los circuitos integrados con diodos leds que disponen de 7 tramos rectos que se iluminan independientemente, sino que

los mismos conceptos son utilizados por las pantallas de cristal líquido o de plasma y similares, que representan visualmente cifras o letras con la utilización de 7 u 8 segmentos.

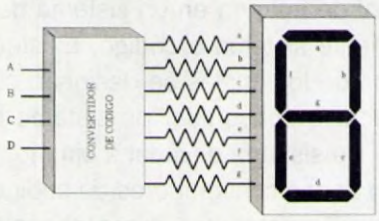


Figura 7.41. Circuito de iluminación del display 7 segmentos

Cada fabricante puede tener sus propios modelos de display 7 segmentos, disponiendo por tanto de tablas de verdad diferentes para el diseño de los mismos. En caso de recurrir a una matriz de puntos que representen las cifras equivalentes del display 7 segmentos, se procederá de similar forma pero teniendo en cuenta que cada punto de un segmento puede tener una consideración independiente.

E_4	E_3	E_2	E_1	S_1-a	S_2-b	S_3-c	S_4-d	S_5-e	S_6-f	S_7-g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	0	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	0	1	1
1	0	1	0	0	0	0	1	1	0	1
1	0	1	1	0	0	1	1	0	0	1
1	1	0	0	0	1	0	0	0	1	1
1	1	0	1	1	0	0	1	0	1	1
1	1	1	0	0	0	0	1	1	1	1
1	1	1	1	0	0	0	0	0	0	0

Tabla 7.14. Iluminación de un display de 7 segmentos

Figura 7.42. Opciones de iluminación

$$S_7=E_3\bar{E}_2+E_4\bar{E}_2+\bar{E}_3E_2+E_2\bar{E}_1 \quad [7.18]$$

$$S_6=E_3\bar{E}_2+E_4\bar{E}_2+E_3\bar{E}_1+\bar{E}_2\bar{E}_1 \quad [7.19]$$

$$S_5=\bar{E}_3\bar{E}_1+E_2\bar{E}_1 \quad [7.20]$$

$$S_4=\bar{E}_3\bar{E}_1+E_2\bar{E}_1+\bar{E}_3E_2+E_3\bar{E}_2E_1 \quad [7.21]$$

$$S_3=\bar{E}_4E_3+\bar{E}_3\bar{E}_2+\bar{E}_3E_1 \quad [7.22]$$

$$S_2=\bar{E}_4\bar{E}_3+\bar{E}_2\bar{E}_1+\bar{E}_3\bar{E}_2 \quad [7.23]$$

$$S_1=\bar{E}_3\bar{E}_2\bar{E}_1+E_4\bar{E}_2E_1+\bar{E}_4E_3E_1+\bar{E}_4\bar{E}_3E_2 \quad [7.24]$$

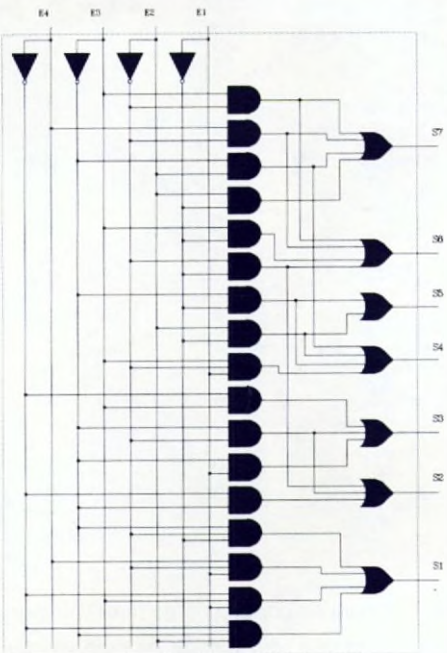


Figura 7.43. Convertidor de código para display 7 segmentos

*## **Ejemplo 7.24**: Obtener la función algebraica que ilumina los números 3, 5 y 7 del display.*

Solución:

Valores binarios de los números 3, 5 y 7: 0011, 0101, 0111

Función algebraica que determina la suma de las tres cifras:

$$F=\bar{A}\bar{B}CD+\bar{A}B\bar{C}D+\bar{A}BCD=\bar{A}CD+\bar{A}BD$$

*## **Ejemplo 7.25**: Diseñar la tabla de verdad de un convertidor de código y un display que ilumine los caracteres 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, b, C, d, E, F.*

Solución:

Figura 7.44

E_4	E_3	E_2	E_1	S_1-a	S_2-b	S_3-c	S_4-d	S_5-e	S_6-f	S_7-g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1

*## **Ejemplo 7.26**: Diseñar un convertidor de código para iluminar un display con 7 segmentos y punto decimal, de tal forma que sólo ilumine los números 0 a 9 y siempre permanezca iluminado el punto decimal.*

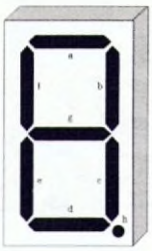


Figura 7.45

Solución:

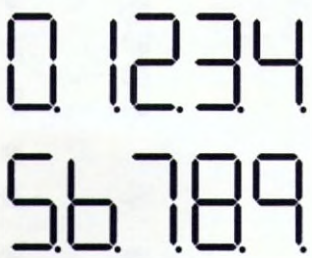


Figura 7.46

La tabla de funcionamiento es la siguiente:

E_4	E_3	E_2	E_1	S_1-a	S_2-b	S_3-c	S_4-d	S_5-e	S_6-f	S_7-g	S_8-h
0	0	0	0	1	1	1	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	1
0	0	1	1	1	1	1	1	0	0	1	1
0	1	0	0	0	1	1	0	0	1	1	1
0	1	0	1	1	0	1	1	0	1	1	1
0	1	1	0	1	0	1	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0	1
1	0	0	0	1	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1	1

Se deducen las siguientes ecuaciones aplicando Karnaugh para funciones especificadas:

$$S_8 = 1$$

$$S_7 = E_4 + E_3\bar{E}_2 + \bar{E}_3E_2 + E_2\bar{E}_1$$

$$S_6 = E_4 + \bar{E}_2\bar{E}_1 + E_3\bar{E}_1 + E_3\bar{E}_2$$

$$S_5 = \bar{E}_3\bar{E}_1 + E_2\bar{E}_1$$

$$S_4 = \bar{E}_3\bar{E}_1 + E_4 + + \bar{E}_3E_2 + E_2\bar{E}_1 + E_3\bar{E}_2E_1$$

$$S_3 = \bar{E}_2 + E_1 + E_3$$

$$S_2 = \bar{E}_3 + \bar{E}_2\bar{E}_1 + E_2E_1$$

$$S_1 = \bar{E}_3\bar{E}_1 + E_4 + E_2 + E_3E_1$$

Estas ecuaciones conducen al siguiente circuito convertidor de código:

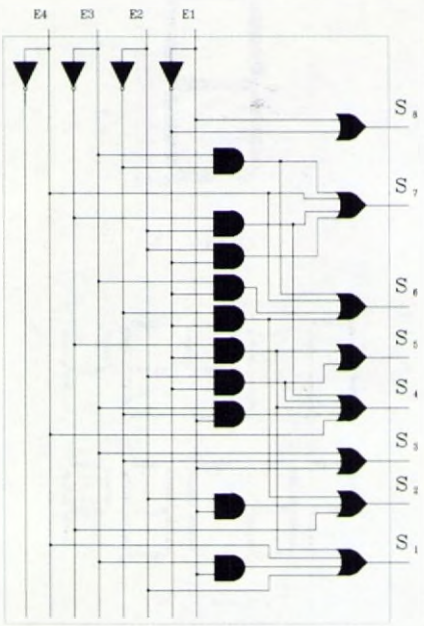


Figura 7.47

7.8. **COMPARADORES**

Un comparador es un circuito que analiza dos cifras binarias indicando la relación de igualdad o desigualdad entre ellas. El circuito combinacional más completo que se puede diseñar tendrá 3 salidas, asignando una a cada opción de la comparación. Así, designando las salidas como **I** (Igual), **M** (Mayor) y **m** (Menor), ante dos entradas binarias el circuito ha de generar una única salida activa según las diferentes posibilidades lógicas de igualdad o desigualdad entre las mismas. Se determina el siguiente criterio de actuación para la definición de la salida activa de dicho circuito:

* Si el bit A es mayor que el B (A>B) se generará un "1" en la salida M
* Si el bit A es menor que el B (A<B) se generará un "1" en la salida m
* Si ambos bits son iguales (A=B) la salida I será un "1"

7.8.1. **Comparador de igualdad-desigualdad**

El circuito más simple de comparación entre dos cifras binarias es el que únicamente determina si ambas son iguales o desiguales. Este circuito denominado comparador de igualdad-desigualdad se basa en el empleo de puertas lógicas XNOR que generan un "1" en caso de igualdad de los bits de entrada y un "0" en caso de desigualdad de los mismos en un solo terminal de salida.

Dependiendo del número de bits que tengan los datos que se hayan de comparar, se dispondrán tantas puertas XNOR de comparación, conectando todas las salidas a través de una puerta AND que determinará la salida del circuito. Puede hacerse un montaje alternativo empleando puertas XOR y una puerta OR de salida si se quiere construir un circuito que genere un "0" en caso de igualdad y un "1" en caso de desigualdad.

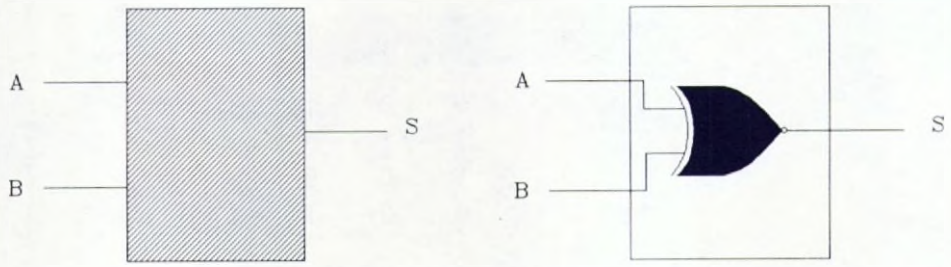


Figura 7.48. Circuito básico de igualdad/desigualdad

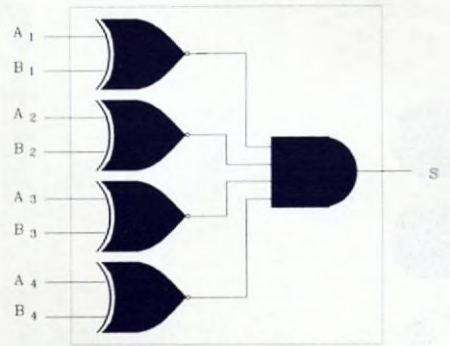


Figura 7.49. Circuito de igualdad/desigualdad para 4 bits (M.1)

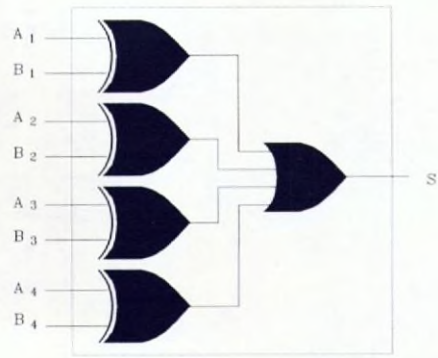


Figura 7.50. Circuito de igualdad/desigualdad para 4 bits (M.2)

7.8.2. **Circuito comparador de un bit**

Se define como un bloque constituido únicamente por 2 entradas de un bit cada una y las tres salidas "M", "m", "I", representado genéricamente en la figura 7.51. La tabla lógica del comparador de un bit es la 7.16.

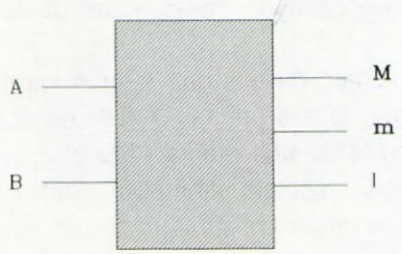


Figura 7.51. Bloque del comparador de un bit

A	B	M	m	I
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

Tabla 7.16. Comparador de un bit

$$M = A\bar{B} \qquad [7.25]$$

$$m = \bar{A}B \qquad [7.26]$$

$$I = \bar{A}\bar{B} + AB = A \odot B \qquad [7.27]$$

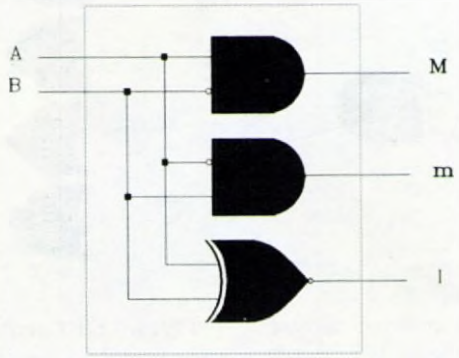


Figura 7.52. Circuito comparador de un bit

7.8.3. **Circuito comparador de dos bits**

El procedimiento de diseño del mismo es similar al realizado para el comparador de un bit, pero su estructura estará formada por comparadores elementales de un bit, conectados a través de un circuito combinacional según se indica en la figura 7.54. Las tablas de verdad de cada uno de los comparadores elementales de un bit responderán a los valores y nomenclaturas de la tabla 7.17 y figuras 7.53 y 7.54, aplicando la tabla base 7.16.

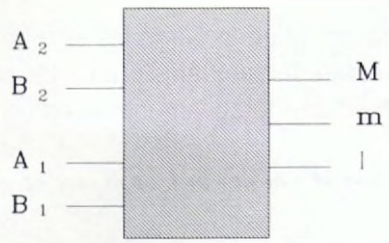


Figura 7.53. Bloque comparador de 2 bits.

A_1	B_1	M_1	m_1	I_1	A_2	B_2	M_2	m_2	I_2
0	0	0	0	1	0	0	0	0	1
0	1	0	1	0	0	1	0	1	0
1	0	1	0	0	1	0	1	0	0
1	1	0	0	1	1	1	0	0	1

Tabla 7.17. Tablas para el comparador de 2 bits

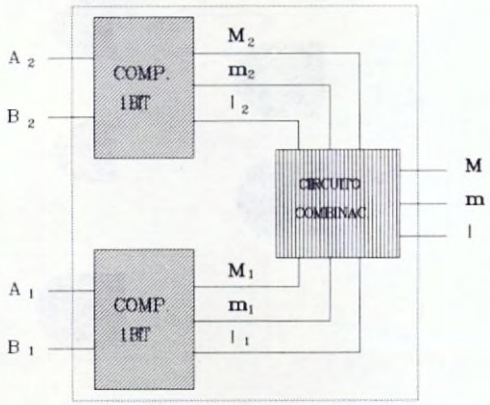


Figura 7.54. Estructura del comparador de 2 bits

La obtención de los valores de salida del comparador pasa por el diseño del circuito combinacional de conexión de los comparadores de un bit, el cuál se obtiene de acuerdo con los siguientes criterios:

a) Para obtener la salida "M" pueden darse las siguientes opciones:
* Si se obtiene M_2 pueden darse cualquiera de las posibilidades M_1, m_1, I_1
* Si se obtiene I_2 debe darse forzosamente la opción M_1

$$M=M_2M_1+M_2m_1+M_2I_1+I_2M_1=M_2+I_2M_1 \qquad [7.28]$$

b) Para obtener la salida "m" pueden darse las siguientes opciones:
* Si se obtiene m_2 pueden darse cualquiera de las posibilidades M_1, m_1, I_1
* Si se obtiene I_2 debe darse forzosamente la opción m_1

$$m=m_2M_1+m_2m_1+m_2I_1+I_2m_1=m_2+I_2m_1 \qquad [7.29]$$

c) Para obtener la salida "I" sólo existe la opción:
* Deben darse simultáneamente I_1 e I_2

$$I=I_1I_2 \qquad [7.30]$$

El circuito combinacional que relaciona los comparadores elementales se representa en la figura 7.55. El circuito completo formado por las puertas lógicas de los comparadores elementales y el circuito combinacional de conexión que definen las tres ecuaciones globales es el representado en la figura 7.56.

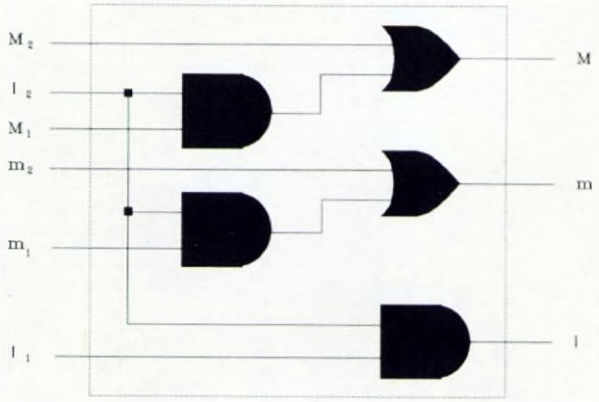


Figura 7.55. Circuito combinacional

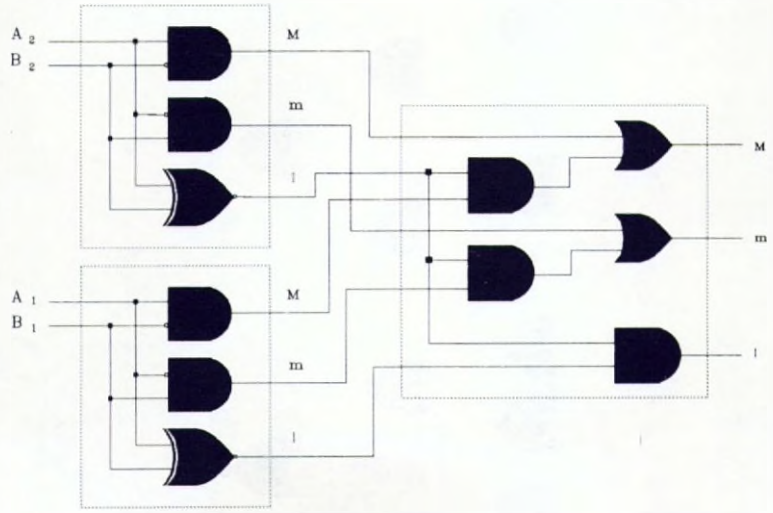


Figura 7.56. Comparador de 2 bits

7.9. **GENERADORES DE PARIDAD**

En la transmisión de información se ha comprobado la necesidad de generar un bit de paridad que detecte los errores. El circuito más adecuado y simple es el que emplea puertas lógicas XOR, ya que genera un "1" a su salida cuando una de las entradas es "1" ó lo que es lo mismo, cuando las entradas son desiguales o impares en el cómputo de "unos" en sus entradas.

En la figura 7.57 se ha representado el generador de paridad para un conjunto de 7 bits a transmitir. Este circuito genera paridad par o impar según la disposición de la salida que se utilice. La paridad impar puede obtenerse mediante negación de la salida de paridad par o bien conectando una puerta XNOR final en vez de la última XOR. La ecuación que define la salida del generador de paridad par es la [7.31] y la del generador de paridad impar la [7.32].

$$B_8 = B_1 \oplus B_2 \oplus B_3 \oplus B_4 \oplus B_5 \oplus B_6 \oplus B_7 \qquad [7.31]$$

$$B_8 = (B_1 \oplus B_2 \oplus B_3 \oplus B_4) \odot (B_5 \oplus B_6 \oplus B_7) \qquad [7.32]$$

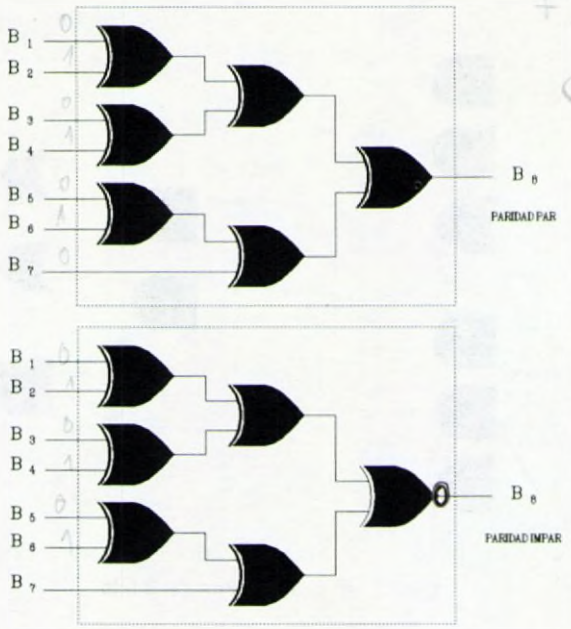


Figura 7.57. Generador de paridad

*## **Ejemplo 7.27**: Aplicar el cronograma de la figura 7.59 al circuito generador de paridad par de 4 bits de la figura 7.58 y obtener gráficamente la salida del mismo.*

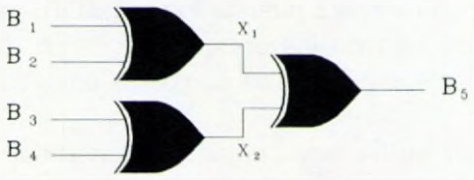


Figura 7.58

Solución:

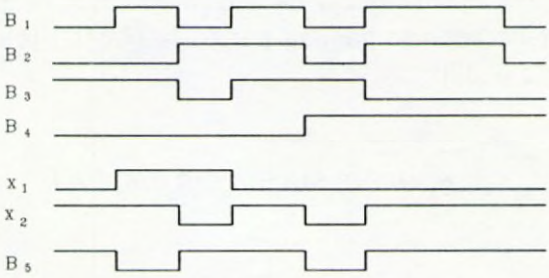


Figura 7.59

*## **Ejemplo 7.28**: Aplicar el cronograma de la figura 7.61 al circuito generador de paridad impar de 7 bits de la figura 7.60.*

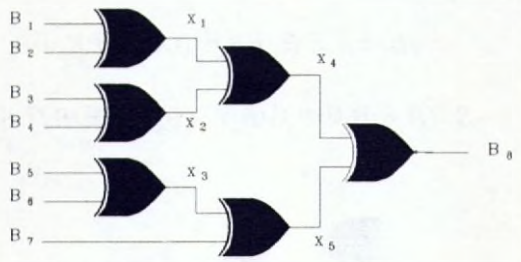


Figura 7.60

Solución:

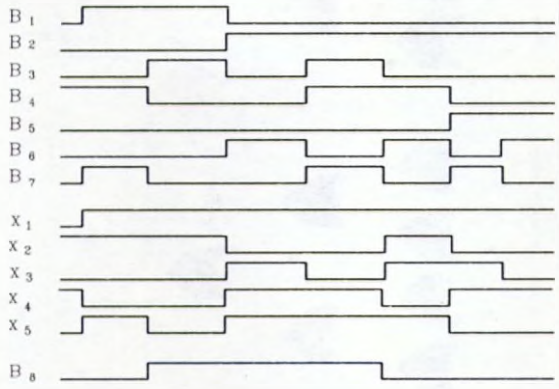


Figura 7.61

7.10. **DETECTORES DE PARIDAD**

Igual que es necesario generar la paridad de la transmisión, se precisa un circuito que detecte si la transmisión efectuada con el bit de paridad es correcta en el punto de recepción. Es necesario por tanto diseñar un detector de paridad, utilizando el mismo criterio que el empleado para los circuitos generadores de paridad.

El circuito correspondiente al detector de paridad para un conjunto de 8 bits se representa en la figura 7.62, perteneciendo 7 de los bits a la información original y el octavo al bit de paridad. Dependiendo de la salida que se

utilice se obtendrá el detector de paridad par o impar, con las mismas consideraciones que en el generador. La ecuación [7.33] define la salida del detector de paridad par y la [7.34] la del detector de paridad impar.

$$S=B_1 \oplus B_2 \oplus B_3 \oplus B_4 \oplus B_5 \oplus B_6 \oplus B_7 \oplus B_8 \qquad [7.33]$$

$$S=(B_1 \oplus B_2 \oplus B_3 \oplus B_4) \odot (B_5 \oplus B_6 \oplus B_7 \oplus B_8) \qquad [7.34]$$

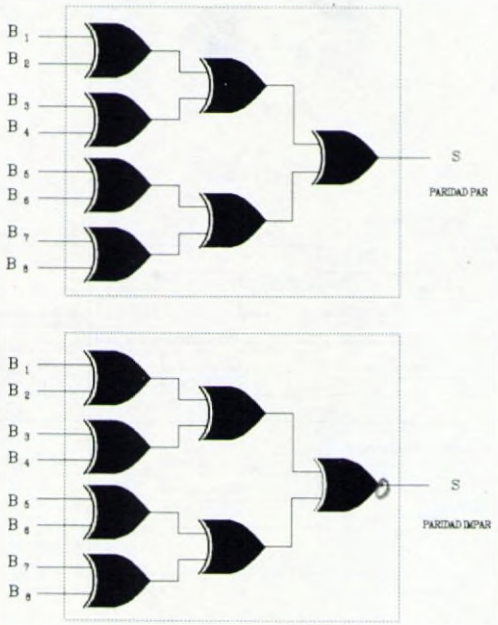


Figura 7.62. Detector de paridad

*## **Ejemplo 7.29**: Aplicar el cronograma de la figura 7.64 al circuito detector de paridad par e 4 bits de la figura 7.63.*

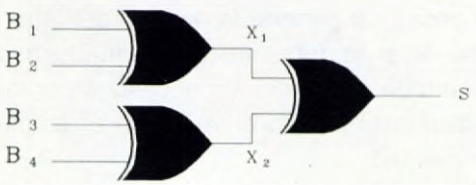


Figura 7.63

Solución:

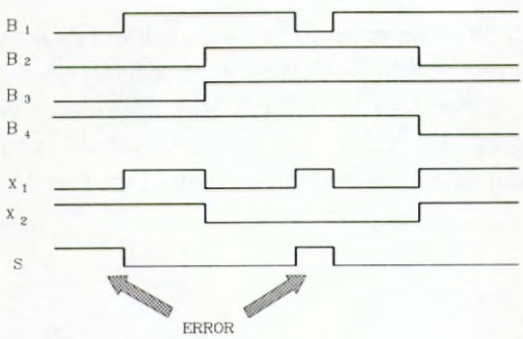


Figura 7.64

***## Ejemplo 7.30**: Aplicar el cronograma de la figura 7.66 al circuito detector de paridad impar de 8 bits de la figura 7.65.*

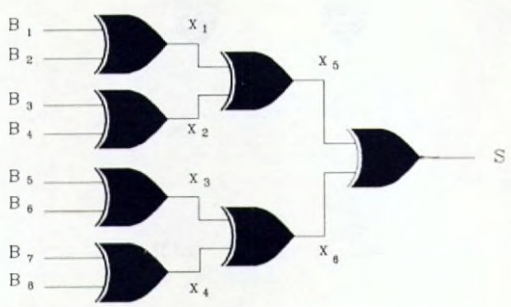


Figura 7.65

Solución:

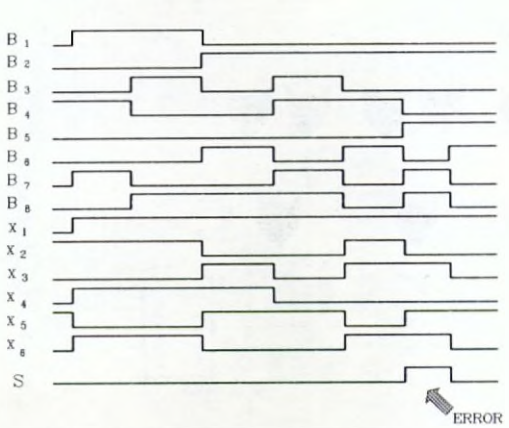


Figura 7.66

7.11. CIRCUITO GENERADOR DE HAMMING

Tomando como base los criterios definidos para la determinación de los bits de paridad necesarios en la detección y corrección de errores mediante el código Hamming, tratado en el tema 3 y representado en el diagrama de bloques de la figura 3.4, se puede confeccionar el circuito generador de Hamming de la figura 7.62. Este circuito añade a los bits de datos originales a

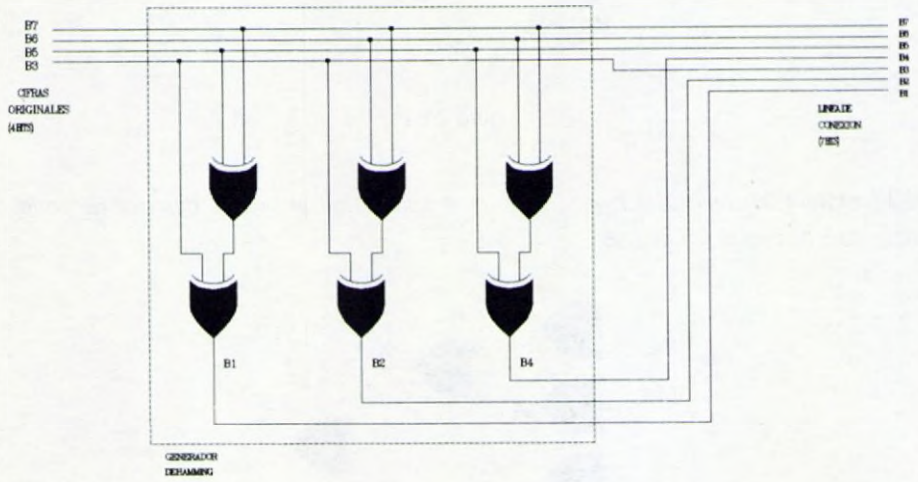


Figura 7.67. Circuito generador de Hamming

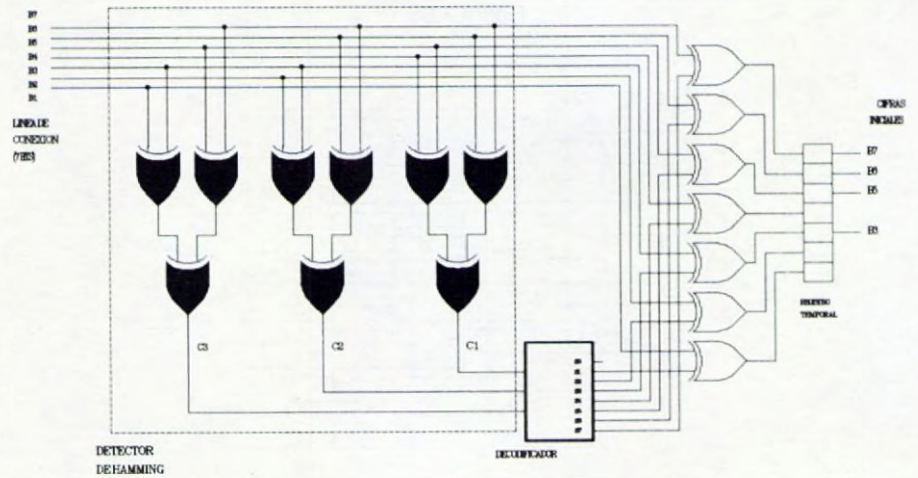


Figura 7.68. Circuito detector de Hamming

transmitir, una serie de bits de paridad que permitirán el control correcto de la transmisión. Para el ejemplo desarrollado de 4 bits se utilizarán 3 bloques generadores de paridad, definidos según el esquema de la figura 7.58 para el supuesto de paridad par.

La línea de transmisión contendrá los 4 bits originales más los 3 bits añadidos por el circuito generador de Hamming. Estos 7 bits transmitidos deben ser chequeados en el equipo receptor por un circuito detector de Hamming como el representado en la figura 7.63. Siguiendo con la misma estructura del equipo transmisor, empleará paridad par y tomará como base de constitución el detector de paridad par de la figura 7.60, aplicado a 4 bits. En caso de detectar error, este circuito procederá a complementar la posición errónea.

TEMA 8
CIRCUITOS ARITMETICOS

8

CIRCUITOS ARITMETICOS

8.1. INTRODUCCION

En este tema se aborda un segundo bloque de circuitos combinacionales, que por abarcar una serie de aplicaciones muy amplias, se analizan independientemente. Son los circuitos aritméticos que realizan operaciones de suma, resta, multiplicación y división. Se analiza el tema partiendo de estructuras simples como las definidas por las tablas de suma y resta que constituyen los semisumadores y semirrestadores. Con ellas se formarán el sumador y el restador completo, analizándose sus circuitos de puertas lógicas.

Estos circuitos simples permitirán mediante diferentes tipos de conexiones, la formación de complejos circuitos para realizar operaciones aritméticas y lógicas. Estos circuitos conducen al diseño y aplicación de la ALU, el operador más completo que realiza dichos cálculos en las unidades centrales de los equipos informáticos. Asimismo, se introducen los circuitos de multiplicación y división que son sustentados por la operación básica de suma mediante conexión de registros y el aprovechamiento de la propiedad de resta mediante complementación del sustraendo.

8.2. ARITMETICA BINARIA

La base de la aritmética binaria son las operaciones de suma y resta. Incluso la operación de resta se realizará a través de la suma, lo que implica que esta última operación y su circuito correspondiente serán la base de funcionamiento y constitución de todos los operadores aritméticos que se diseñan.

8.2.1. **Suma binaria**

La operación de suma de 2 cifras binarias se realiza sumando los bits correspondientes a los coeficientes de iguales potencias. Esta operación de suma de 2 bits, denominados augendo y adendo, puede dar lugar a la obtención de un valor de suma y de una nueva cifra denominada acarreo, que ha de añadirse al resultado de la suma de los dos bits de la potencia inmediata superior, con lo cuál en estos términos, se realizaría una suma de tres bits. La tabla 8.1 correspondiente a la suma de 2 bits A y B, proporciona S como resultado de la suma y A_c como el acarreo que se obtiene y que se podrá añadir a otra etapa sumadora.

A	B	S	A_c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Tabla 8.1. Suma binaria

Cuando se emplean registros de un equipo de proceso de datos para realizar la operación de suma, el número de bits está limitado a la capacidad máxima de almacenamiento del registro. Teniendo en cuenta que el bit más significativo se destina al bit de signo, puede ocurrir que al realizar la suma de dos datos almacenados en registros se supere la capacidad máxima de almacenamiento. En estos casos se produce desbordamiento (overflow) en el sistema y el resultado que se obtiene no es utilizable.

*## **Ejemplo 8.1**: Realizar la suma binaria de los números 1011011,010 y 11101010,10:*

Solución:

```
    1011011,010                          91,25
+   11101010,10                      +   234,5
-------------                        ----------
   10110001,110   Suma previa           325,75
+    1 1 1 ,      Acarreo
-------------
  101000101,110   Suma final
```

*## **Ejemplo 8.2**: Realizar la suma binaria de los números 11111011011,10101 y 100001010,0101:*

Solución:

```
   11111011011,10111                          2011,71875
+    100001010,0101                       +     266,3125
-------------------                        ---------------
   11011010001,11101   Suma previa          2278,03125
+      1  11 , 1       Acarreo
-------------------
 1000111100110,00001   Suma final
```

*## **Ejemplo 8.3**: Realizar la suma binaria de los números 1111011011011 y 110000101000 usando bit de signo (sin desglosar en suma previa y acarreo):*

Solución:

```
  0 01111011011011
+ 0 00110000101000
-------------------
  0 10101100000011   Suma final
```

*## **Ejemplo 8.4**: Realizar la suma binaria de los números 111110110110111 y 1110000101010l almacenados en registros de 16 bits:*

Solución:

```
  0 111110110110111
+ 0 011100001010101
-------------------
  1 011011000001100   Desbordamiento
```

8.2.2. **Resta binaria**

La realización de la operación de resta de dos cifras binarias, es equivalente a la de suma, pero teniendo en cuenta que los equivalentes al augendo y adendo se denominan ahora minuendo y sustraendo y el acarreo ahora es préstamo. La tabla de resta de dos bits es ahora la expresada por 8.2. Para realizar este tipo de operación es necesario que el minuendo sea mayor que el sustraendo (En el apartado 8.3 se analizan todas las posibilidades de resta).

A	B	R	P
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Tabla 8.2. Resta binaria

*## **Ejemplo 8.5**: Realizar la resta binaria de los números 1011011,101 y 101010,01:*

Solución:

```
   1011011,101                                  91,625
 -   101010,01                               -  42,25
 ------------                                 -------
   1110001,111    Resta previa                 49,375
 -    1 1 ,1     Préstamo
 ------------
   0110001,011    Resta final
```

*## **Ejemplo 8.6**: Realizar la resta binaria de los números 11111011011,1011 y 100001010,0101:*

Solución:

```
   11111011011,10111                            2011,71875
 -   100001010,0101                           -  266,3125
 ---------------                              -----------
   11011010001,11101   Resta previa            1745,40525
 -             ,1      Préstamo
 ---------------
   11011010001,01101   Resta final
```

*## **Ejemplo 8.7**: Realizar la resta binaria de los números 111110110110111 y 1001010100101 con bit de signo (sin desglosar en resta previa y préstamo):*

Solución:

```
     0 111110110110111
   - 0   1001010100101
   -------------------
     0 110101100010010   Resta final
```

8.2.3. **Circuito semisumador**

Cuando se definió el álgebra de Boole se estudiaron las operaciones aritméticas, entre las que se incluía la de suma binaria. Este apartado tarta del diseño del circuito físico que realiza dicha operación de suma. Teniendo en cuenta la tabla de suma 8.1, se obtienen las ecuaciones [8.1] y [8.2] de definición del circuito de puertas lógicas. El esquema representado en la figura 8.1 es el que se denomina semisumador, dado que no se considera el acarreo que pueda proceder de una etapa anterior de suma en el caso de que se utilicen varios semisumadores en cascada.

$$S=\overline{A}B+A\overline{B}=A\oplus B \qquad [8.1]$$

$$A_c=AB \qquad [8.2]$$

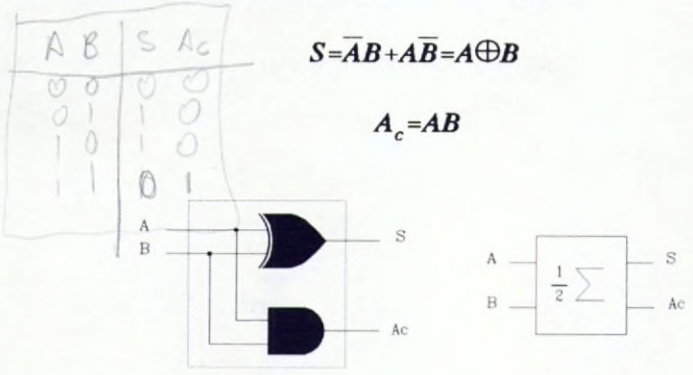


Figura 8.1. Semisumador

8.2.4. **Sumador completo**

Cuando a un semisumador se le añade un posible acarreo de una operación previa, es decir, se suman 3 bits, se obtiene el denominado "Sumador Completo" ó simplemente "Sumador". Su tabla de verdad 8.3 analiza A y B como los bits a sumar y A_p el acarreo previo que se considera el tercer bit de suma. S sigue siendo el resultado final de la suma y A_f el acarreo final que se produce. El circuito correspondiente a dichas ecuaciones se representa en la figura 8.2, agrupando las puertas lógicas en forma de dos semisumadores conectando sus acarreos a través de una puerta OR. El esquema bloque del sumador completo es el indicado en la figura 8.3.(a), y el que se utilizará más adelante en los diagramas aritméticos es el indicado en la figura 8.3.(b).

$$S=\overline{A}A_p\overline{B}+\overline{A}B\overline{A_p}+\overline{A_p}A\overline{B}+ABA_p \qquad [8.3]$$

$$A_f=\overline{A}BA_p+A\overline{B}A_p+AA_p\overline{B}+ABA_p \qquad [8.4]$$

A	B	A_p	S	A_f
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tabla 8.3. Sumador completo

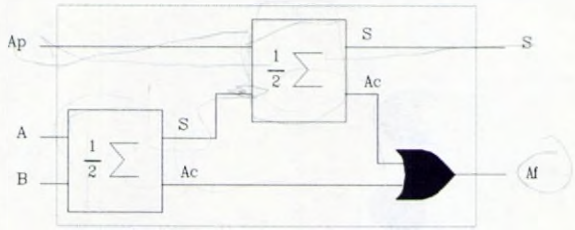


Figura 8.2. Sumador completo

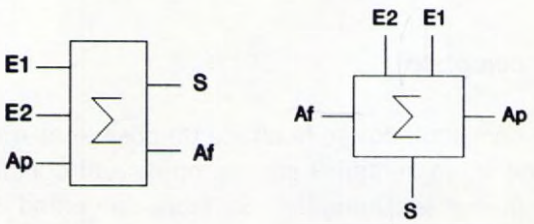


Figura 8.3. Diagramas bloque del sumador completo

La realización de una suma de varios bits supone la conexión en cascada de varios sumadores elementales como los representados en la figura 8.4, donde los acarreos se transmiten al sumador de orden superior. El esquema de la figura 8.4, se simplifica en un solo bloque integrado, según se representa en la figura 8.5, donde cada pareja de datos de entrada se corresponde con una salida y los acarreos intermedios quedan internos en el circuito.

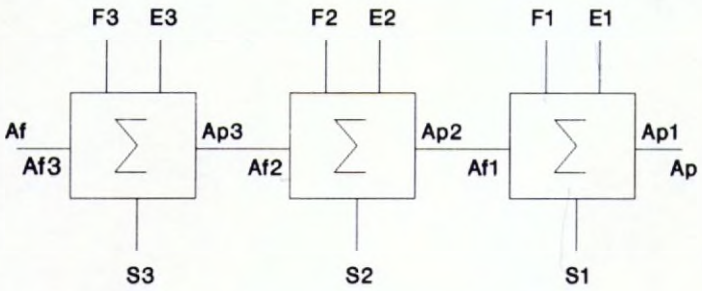


Figura 8.4. Sumador de cifras binarias de 3 bits

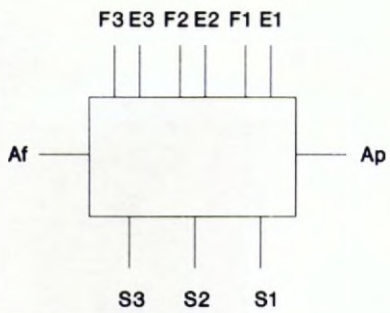


Figura 8.5. Esquema simplificado del sumador binario

*## **Ejemplo 8.8**: Aplicar el cronograma de la figura 8.6 a las entradas del circuito semisumador y obtener la salida.*

Solución:

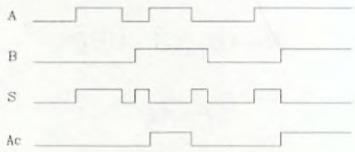


Figura 8.6

*## **Ejemplo 8.9**: Aplicar el cronograma de la figura 8.7 a las entradas del circuito sumador completo y obtener la salida.*

Solución:

Figura 8.7

***## Ejemplo 8.10**: Aplicar el cronograma de la figura 8.8 a las entradas del circuito sumador completo y obtener la salida.*

Solución:

Figura 8.8

8.2.5. **Circuito semirrestador**

Al igual que para el semisumador, se puede obtener el semirrestador partiendo de la tabla simple de resta de dos bits representada en 8.2. El nombre de semirrestador se le asigna por idénticos motivos que para el semisumador.

$$R=\overline{A}B+A\overline{B}=A\oplus B \qquad [8.5]$$

$$P=\overline{A}B \qquad [8.6]$$

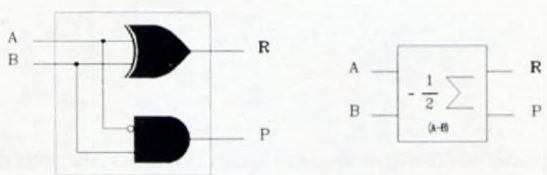


Figura 8.9. Semirrestador

8.2.6. **Restador completo**

Cuando un semirrestador dispone también del préstamo de una etapa previa, se manipulan 3 bits en cada operación y se obtiene el restador completo. Su tabla de verdad es la 8.4. El préstamo final debe obtenerse mediante los préstamos de las dos etapas semirrestadoras, conectando ambas a través de una puerta sumadora. El esquema completo que se obtiene se representa en la figura 8.10. Los bloques acoplados en cascada para realizar una operación de varios bits, se conectan de idéntica forma que para el sumador completo, sustituyendo en este caso el acarreo previo por el préstamo previo, con lo que se obtendrán varias restas parciales y un préstamo final.

A	B	P_p	R	P_f
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Tabla 8.4. Restador completo

$$R=\overline{A}P_p\overline{B}+\overline{A}B\overline{P_p}+\overline{P_p}A\overline{B}+ABP_p \qquad [8.7]$$

$$P_f=\overline{A}P_p+\overline{A}B+BP_p \qquad [8.8]$$

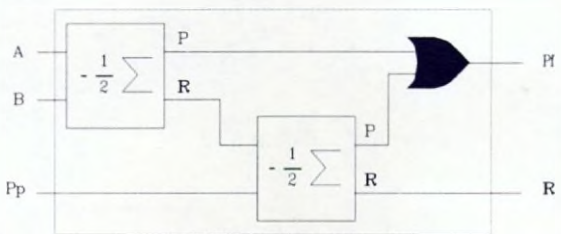


Figura 8.10. Restador completo

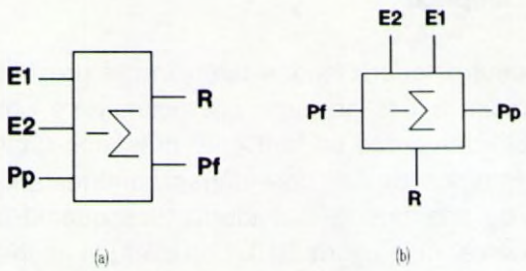


Figura 8.11. Bloque restador completo

*## **Ejemplo 8.11**: Aplicar el cronograma de la figura 8.12 a las entradas del semirrestador obteniendo la salida gráficamente.*

Solución:

Figura 8.12

*## **Ejemplo 8.12**: Aplicar el cronograma de la figura 8.13 a las entradas del restador completo, obteniendo la salida gráficamente.*

Solución:

Figura 8.13

*## **Ejemplo 8.13**: Aplicar el cronograma de la figura 8.14 a las entradas del restador completo, obteniendo la salida gráficamente.*

Solución:

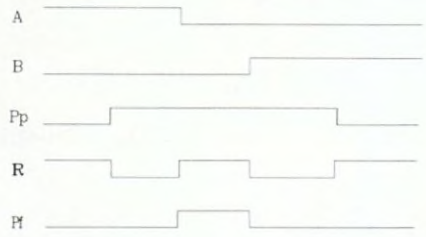


Figura 8.14

8.3. **COMPLEMENTOS**

La operación de resta binaria, puede hacerse equivalente a una operación de suma binaria. Para ello, es necesario definir los conceptos de complemento a 1 y complemento a 2, que intervendrán en la transformación de la resta en suma. Estos complementos se utilizan para expresar de otra forma únicamente los números negativos que hasta ahora se indicaban con un "1" en el bit de signo y la expresión binaria correspondiente y poder utilizarlos en los cálculos aritméticos. De esta forma los números negativos utilizados en los cálculos seguirán teniendo el bit de signo "1" pero el resto de los bits tomarán valores diferentes según el tipo de complemento a utilizar.

8.3.1. **Complemento a "1"**

Cualquier número binario puede transformarse en su complemento a 1, mediante la diferencia entre la potencia de base dos inmediatamente superior al valor de dicho número y la suma de dos términos, el primero de ellos el número binario y el segundo la potencia -f de base dos (De aplicación sólo a números negativos). La aplicación de la fórmula [8.9] se puede realizar tanto con números en decimal como en binario. Se puede observar además que el complemento a 1 es equivalente al cambio de ceros por unos y unos por ceros del número binario B. Si se emplea bit de signo, se considera como un bit más incluido en el número a complementar.

$$C_B^1 = 2^e - (B + 2^{-f}) \qquad [8.9]$$

B=Valor absoluto del número decimal o binario a procesar
e=Número de bits de la parte entera de B
f=Número de bits de la parte fraccionaria de B

*## **Ejemplo 8.14**: Calcular el complemento a 1 del número decimal -219,375.*

Solución:

$$B=219{,}375_{10}=11011011{,}011_2$$

$$C_B^1=2^8-(219{,}375+2^{-3})=36{,}5_{10}=100100{,}1_2$$

*## **Ejemplo 8.15**: Calcular el complemento a 1 del número decimal -76,25 realizando únicamente operaciones binarias.*

Solución:

$$B=76{,}25_{10}=1001100{,}01_2$$

$$C_B^1=10000000-(1001100{,}01+0{,}01)=00110011{,}10$$

*## **Ejemplo 8.16**: Calcular el complemento a 1 del número decimal -97 realizando únicamente operaciones binarias.*

Solución:

$$B=97_{10}=1100001_2$$

$$C_B^1=10000000-(1100001+1)=00011110$$

*## **Ejemplo 8.17**: Obtener sin necesidad de cálculo aritmético el complemento a 1 del número binario -110010010,1101.*

Solución:

$$B=110010010.1101$$

$$C_B^1=001101101,0010$$

*## **Ejemplo 8.18**: Obtener sin necesidad de cálculo aritmético el complemento a 1 del número binario -111110010011.*

Solución:

$$B=111110010011$$

$$C_B^1=000001101100$$

8.3.2. **Complemento a "2"**

Igual que para el complemento a 1, cualquier número binario puede transformarse en su complemento a 2 mediante la diferencia entre la potencia de base 2 inmediatamente superior al valor del número y dicho número. El complemento a 2 es el resultado de realizar el complemento a 1 en el número binario B, añadiéndole un "1" a la cifra situada más a la derecha (la menos significativa). Igualmente el bit de signo se puede utilizar incluyéndolo en el total de bits a complementar.

$$C_B^2 = 2^e - B \qquad [8.10]$$

B=Valor absoluto del número decimal o binario a procesar
e=Número de bits de la parte entera de B

*## **Ejemplo 8.19**: Calcular el complemento a 2 del número decimal -219,375.*

Solución:

$$B = 219{,}375_{10} = 11011011{,}011_2$$

$$C_B^2 = 2^8 - 219{,}375 = 36{,}625_{10} = 100100{,}101_2$$

*## **Ejemplo 8.20**: Calcular el complemento a 2 del número decimal -76,25 realizando únicamente operaciones binarias.*

Solución:

$$B = 76{,}25_{10} = 1001100{,}01_2$$

$$C_B^2 = 10000000 - 1001100{,}01 = 00110011{,}11$$

*## **Ejemplo 8.21**: Calcular el complemento a 2 del número decimal -86.*

Solución:

$$B = 86_2 = 1010110_2$$

$$C_B^2 = 10000000 - 1010110 = 00101010$$

*## **Ejemplo 8.22**: Obtener sin necesidad de cálculo aritmético el complemento a 2 del número binario -1010010010,1011.*

Solución:

$$B = 101010010{,}1011$$

$$C_B^2 = 010101101{,}0101$$

8.4. **RESTA BINARIA USANDO COMPLEMENTOS**

La operación de resta binaria, se indicó anteriormente que podía hacerse equivalente a una suma binaria. Si se sustituye el sustraendo de la operación de resta por el complemento a 1 ó a 2, se puede entonces realizar la suma que equivale a la operación de resta. Esto es sumamente importante, especialmente en los casos de tener un sustraendo de valor absoluto superior al minuendo, operación de resta que no se puede realizar al producirse desbordamiento.

Para comprender mejor la acción de complementar, se supondrán inicialmente los dos números minuendo y sustraendo con el bit de signo "0" previamente al complemento aunque el número a utilizar sea negativo. Al realizar dicha operación se complementa también el bit de signo convirtiéndose en "1". Si posteriormente a esta cifra complementada se le vuelve a aplicar complemento, se utilizará el nuevo bit de signo "1" obtenido, convirtiéndose en "0" después de la segunda complementación. Una operación de resta, según el signo de los términos que se empleen, podrá tomar cualquiera de las cuatro configuraciones siguientes:

a) Minuendo y sustraendo positivos:

$$\begin{array}{r} A \\ -\ B \\ \hline C \end{array} = + \begin{array}{r} A \\ (-B) \\ \hline C \end{array} = + \begin{array}{r} A \\ (Cto.B) \\ \hline C \end{array}$$

b) Minuendo negativo y sustraendo positivo:

$$\begin{array}{r} (-A) \\ -\ B \\ \hline C \end{array} = + \begin{array}{r} (-A) \\ (-B) \\ \hline C \end{array} = + \begin{array}{r} (Cto.A) \\ (Cto.B) \\ \hline C \end{array}$$

c) Minuendo positivo y sustraendo negativo:

$$\begin{array}{r} A \\ -\ (-B) \\ \hline C \end{array} = + \begin{array}{r} A \\ (-(-B)) \\ \hline C \end{array} = + \begin{array}{r} A \\ (Cto.(Cto.B)) \\ \hline C \end{array} = + \begin{array}{r} A \\ B \\ \hline C \end{array}$$

d) Minuendo y sustraendo negativos:

$$\begin{array}{r} (-A) \\ -\ (-B) \\ \hline C \end{array} = \begin{array}{r} (Cto.A) \\ +(Cto.(Cto.B)) \\ \hline C \end{array} = + \begin{array}{r} (Cto.A) \\ B \\ \hline C \end{array}$$

Cuando se utilizan registros de un sistema digital, puede producirse desbordamiento en las opciones (b) y (c). Se obtendría en valor absoluto una cantidad mayor que la máxima admisible por el registro, lo que determina que el resultado final no es utilizable. Se detecta el desbordamiento en estas situaciones siempre que el bit de signo del resultado final sea diferente que los bits de signo de los términos a sumar. En los microprocesadores se realiza este control del desbordamiento utilizando una puerta XOR cuyas entradas son el acarreo final y el acarreo del bit anterior al más significativo (el bit de signo). Cuando sólo se produce uno de estos dos acarreos se produce desbordamiento y cuando se obtienen los dos, el resultado es correcto.

8.4.1. **Resta binaria con complemento a 1**

Para realizar esta operación, se sustituye el sustraendo por su complemento a 1, teniendo en cuenta que el número de bits se ampliará si fuera necesario hasta obtener los mismos que el minuendo, es decir, se sumarán siempre igual número de coeficientes. Si en la última operación de suma se produce acarreo, **EL ACARREO FINAL DEBE SUMARSE AL BIT MENOS SIGNIFICATIVO**.

Esta condición del acarreo final, se cumple incluso para los casos en que se emplee el bit de signo. En estos casos, el acarreo de la suma se añadirá al resultado de la suma de los bits de signo, y si se produjera acarreo, éste se sumaría al bit menos significativo. Si no se emplea bit de signo el resultado será **POSITIVO SI EXISTE ACARREO FINAL**, y será **NEGATIVO SI NO EXISTE ACARREO FINAL**. Si se emplea bit de signo el resultado final será positivo o negativo según el valor que tenga dicho bit después de operar y de acuerdo con el criterio establecido para el mismo. (La única salvedad que tiene esta regla es para el caso de sumas con bit de signo en las opciones (b) y (c), en las que se atendrá al criterio fijado por el bit de signo).

SI EL RESULTADO ES NEGATIVO LA EXPRESION BINARIA OBTENIDA ESTA EN COMPLEMENTO A 1. Para obtener por tanto el número definitivo es necesario quitar el complemento a 1 a dicha resta final volviendo a complementar. Conviene tener en cuenta que si el complemento a 1 se realiza cambiando ceros por unos y unos por ceros, se deben transformar todos los bits, incluso los que se hayan añadido para equiparar el número de coeficientes.

*## **Ejemplo 8.23**: Realizar la resta en complemento a 1 de los números decimales 176,25 y 123,625.*

Solución:

$M = 176{,}25_{10} = 10110000{,}010_2$

$S = 123{,}625_{10} = 01111011{,}101_2$

$C_S^1 = 10000100{,}010$

	10110000,010	M
+	10000100,010	Cto. 1
	00110100,100	Resta parcial
+	1	Acarreo
	00110100,101	Resta final

*## **Ejemplo 8.24**: Realizar la resta en complemento a 1 de los números decimales 123,625 y 176,25.*

Solución:

$M = 123{,}625_{10} = 01111011{,}101_2$

$S = 176{,}25_{10} = 10110000{,}010_2$

$C_S^1 = 01001111{,}101$

	01111011,101	M
+	01001111,101	Cto. 1
	11001011,010	Resta en cto. 1
-	00110100,101	Resta final

*## **Ejemplo 8.25**: Realizar la resta en complemento a 1 de los números decimales 196,5 y 23,625.*

Solución:

$M = 196{,}5_{10} = 11000100{,}100_2$

$S = 23{,}625_{10} = 00010111{,}101_2$

$C_S^1 = 11101000{,}010$

	11000100,100	M
+	11101000,010	Cto. 1
	10101100,110	Resta parcial
+	1	Acarreo
	10101100,111	Resta final

*## **Ejemplo 8.26**: Realizar la resta en complemento a 1 de los números decimales 23,625 y 196,5.*

Solución:

$M=23{,}625_{10}=00010111{,}101_2$

$S=196{,}5_{10}=11000100{,}100_2$

$C_S^1=00111011{,}011$

	00010111,101	M
+	00111011,011	Cto. 1
	01010011,000	Resta en cto. 1
-	10101100,111	Resta final

*## **Ejemplo 8.27**: Realizar la resta binaria utilizando complemento a 1 de los números -126 y +20 almacenados en registros de 8 bits.*

Solución:

$M=-126_{10}=11111110_2$

$C_M^1=10000001$

$S=20_{10}=00010100_2$

$C_S^1=11101011$

	1 0000001	Cto. 1 de M
+	1 1101011	Cto. 1 de S
	0 1101101	Desbordamiento

*## **Ejemplo 8.28**: Realizar la resta binaria utilizando complemento a 1 de los números -126 y -20 almacenados en registros de 8 bits.*

Solución:

$M=-126_{10}=11111110_2$

$C_M^1=10000001$

$S=-20_{10}=10010100_2$

$C_S^1=11101011$

$C_{Cto.1}^1=00010100$

	1 0000001	Cto. 1 de M
+	0 0010100	Cto. 1 de Cto. 1 de S
	1 0010101	Resta en cto. 1
	1 1101010	Resta final

*## **Ejemplo 8.29**: Realizar la resta binaria utilizando complemento a 1 de los números +126 y -20 almacenados en registros de 8 bits.*

Solución:

$M=126_{10}=01111110_2$

$S=-20_{10}=10010100_2$

$C_S^1=11101011$

$C_{Cto.1}^1=00010100$

```
  0 1111110   M
+ 0 0010100   Cto. 1 de Cto. 1 de S
-----------
  1 0010010   Desbordamiento
```

*## **Ejemplo 8.30**: Realizar la resta binaria utilizando complemento a 1 de los números +126 y -20 almacenados en registros de 16 bits.*

Solución:

$M=126_{10}=0000000001111110_2$

$S=-20_{10}=1000000000010100_2$

$C_S^1=1111111111101011$

$C_{Cto.1}^1=0000000000010100$

```
  0 000000001111110   M
+ 0 000000000010100   Cto. 1 de Cto. 1 de S
-------------------
  0 000000010010010   Resta final
```

*## **Ejemplo 8.31**: Realizar la resta binaria utilizando complemento a 1 de los números -126 y +20 almacenados en registros de 16 bits.*

Solución:

$M=-126_{10}=1000000001111110_2$

$C_M^1=1111111110000001$

$S=20_{10}=0000000000010100_2$

$C_S^1=1111111111101011$

```
  1 111111110000001   M
+ 1 111111111101011   Cto. 1 de S
-------------------
  1 111111101101100   Resta previa
+                 1   Acarreo
-------------------
  1 111111101101101   Resta en cto. 1
-------------------
  1 000000010010010   Resta final
```

8.4.2. **Resta binaria con complemento a 2**

Para realizar esta operación, se sustituye el sustraendo por su complemento a 2, teniendo en cuenta que el número de bits se ampliará si fuera necesario hasta obtener el mismo número que el minuendo, es decir, se sumarán siempre igual número de coeficientes. Si en el complemento a 1 se utilizaba el acarreo final, en el complemento a 2 **EL ACARREO FINAL SE DESPRECIA.**

Esta condición del acarreo final, se cumple incluso cuando se emplee el bit de signo. En estos casos, el acarreo de la suma se añadirá a la suma de los bits de signo, y si se produjera acarreo, éste se despreciaría. Si no se emplea bit de signo el resultado será **POSITIVO SI EXISTE ACARREO FINAL**, y será **NEGATIVO SI NO EXISTE ACARREO FINAL**. Si se emplea bit de signo se aplicará el criterio correspondiente al mismo.

SI EL RESULTADO ES NEGATIVO LA EXPRESION BINARIA OBTENIDA ESTA EN COMPLEMENTO A 2. Para obtener por tanto el número definitivo es necesario complementar dicha cifra. Se debe tener en cuenta la misma observación que se hizo para el complemento a 1 en cuanto a cambiar ceros por unos y unos por ceros en todos los bits, incluso los que se hayan añadido para equiparar el número de coeficientes y lo relativo al complemento del bit de signo y transformación de resta en suma. Todos los aspectos analizados en el complemento a 1 para el desbordamiento son igualmente de aplicación al complemento a 2.

*## **Ejemplo 8.32**: Realizar la resta en complemento a 2 de los números decimales 176,25 y 123,625.*

Solución:

$M = 176{,}25_{10} = 10110000{,}010_2$

$S = 123{,}625_{10} = 01111011{,}101_2$

$C_S^2 = 10000100{,}011$

	10110000,010	M
+	10000100,011	Cto. 2
	00110100,101	Resta final

*## **Ejemplo 8.33**: Realizar la resta en complemento a 2 de los números decimales 123,625 y 176,25.*

Solución:

$M=123{,}625_{10}=01111011{,}101_2$

$S=176{,}25_{10}=10110000{,}010_2$

$C_S^2=01001111{,}110$

01111011,101	M	
+ 01001111,110	Cto. 2	
11001011,011	Resta en cto. 2	
- 00110100,101	Resta final	

*## **Ejemplo 8.34**: Realizar la resta en complemento a 2 de los números decimales 196,5 y 23,625.*

Solución:

$M=196{,}5_{10}=11000100{,}100_2$

$S=23{,}625_{10}=00010111{,}101_2$

$C_S^2=11101000{,}011$

11000100,100	M	
+ 11101000,011	Cto. 2	
10101100,111	Resta final	

*## **Ejemplo 8.35**: Realizar la resta en complemento a 2 de los números decimales 23,625 y 196,5.*

Solución:

$M=23{,}625_{10}=00010111{,}101_2$

$S=196{,}5_{10}=11000100{,}100_2$

$C_S^2=00111011{,}100$

00010111,101	M	
+ 00111011,100	Cto. 2	
01010011,001	Resta en cto. 2	
- 10101100,111	Resta final	

*## **Ejemplo 8.36**: Realizar la resta en complemento a 2 en un registro de 8 bits de los números decimales -126 y +84.*

Solución:

$M=-126_{10}=11111110_2$

$C^2_M=10000010$

$S=84_{10}=01010100_2$

$C^2_S=10101100$

```
  1 0000010   Cto. 2 de M
+ 1 0101100   Cto. 2 de S
-----------
  0 0101110   Desbordamiento
```

*## **Ejemplo 8.37**: Realizar la resta binaria utilizando complemento a 2 de los números -126 y -84 almacenados en registros de 8 bits.*

Solución:

$M=-126_{10}=11111110_2$

$C^2_M=10000010$

$S=-84_{10}=11010100_2$

$C^2_S=10101100$

$C^2_{Cto.2}=01010100$

```
  1 0000010   Cto. 2 de M
+ 0 0010100   Cto. 2 de Cto. 2 de S
-----------
  1 0010110   Resta en cto. 2
-----------
  1 1101010   Resta final
```

*## **Ejemplo 8.38**: Realizar la resta binaria utilizando complemento a 2 de los números +126 y -84 almacenados en registros de 8 bits.*

Solución:

$M=126_{10}=01111110_2$

$S=-84_{10}=11010100_2$

$C^2_S=10101100$

$C^2_{Cto.2}=01010100$

```
  0 1111110   M
+ 0 0010100   Cto. 2 de Cto. 2 de S
-----------
  1 0010010   Desbordamiento
```

*## **Ejemplo 8.39**: Realizar la resta binaria utilizando complemento a 2 de los números +126 y -84 almacenados en registros de 16 bits.*

Solución:

$M=126_{10}=0000000001111110_2$

$S=-84_{10}=1000000001010100_2$

$C_S^2=1111111110101100$

$C_{Cto.2}^2=0000000001010100$

```
  0 000000001111110  M
+ 0 000000001010100  Cto. 2 de Cto. 2 de S
-------------------
  0 000000011010010  Resta final
```

*## **Ejemplo 8.40**: Realizar la resta binaria utilizando complemento a 2 de los números -126 y +84 almacenados en registros de 16 bits.*

Solución:

$M=-126_{10}=1000000001111110_2$

$C_M^2=1111111110000010_2$

$S=84_{10}=0000000001010100_2$

$C_S^2=1111111110101100$

```
  1 111111110000010  M
+ 1 111111111101100  Cto. 2 de S
-------------------
  1 111111101101110  Resta en cto. 2
-------------------
  1 000000010010010  Resta final
```

8.5. CIRCUITO DE SUMA BINARIA

El circuito de suma binaria se puede suponer formado por sumadores elementales de un bit conectados en cascada a través de los acarreos de entrada y salida tal como se indica en la figura 8.15. Cada uno de estos sumadores elementales es un sumador completo ya que realiza la suma de dos bits de entrada y de un tercer bit de acarreo de la etapa precedente.

Este esquema completo puede hacerse tan grande como se quiera conectando los sumadores elementales que se requieran, o mediante la conexión de bloques sumadores estandar en circuitos integrados normalmente de cuatro bits. En la figura 8.16 se representa el esquema de un sumador binario de 16 bits, formado por 4 sumadores binarios genéricos de 4 bits, constituido

cada uno de ellos por los elementos descritos en la figura 8.15. En circuitos más avanzados, especialmente operadores de unidades de proceso, se utiliza un generador rápido de acarreo. Consiste en un circuito combinacional que tomando como entradas las salidas de acarreo de las distintas etapas sumadoras proporciona una salida de acarreo mucho más rápida que si se espera a que el circuito vaya reprocesando mediante nuevos ciclos de suma cada uno de los acarreos que se vayan generando al sumar.

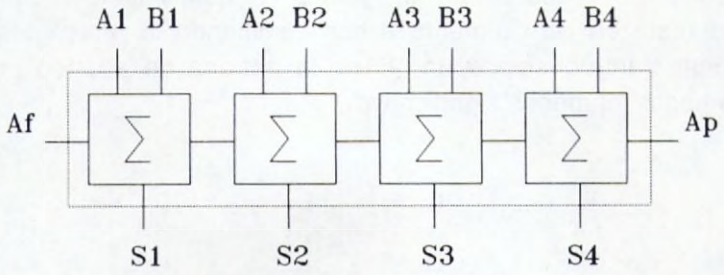


Figura 8.15. Sumador en cascada

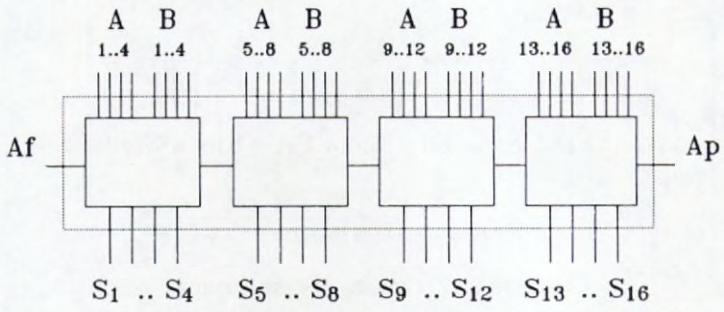


Figura 8.16. Sumador binario de 16 bits

8.6. **CIRCUITO DE RESTA BINARIA**

El circuito de resta binaria es similar al descrito para la suma binaria. Sin embargo, lo interesante de la operación de resta binaria no es la descripción del circuito de resta sino la posibilidad de efectuarla mediante circuitos de suma binaria. Una operación de resta binaria puede ser realizada a través de la suma mediante el complemento a uno o dos del sustraendo. Dependiendo del tipo de complemento se tendrán o no en cuenta los aca-

rreos finales para operar con ellos. Así por ejemplo, el circuito de la figura 8.17 podrá realizar la resta binaria si las entradas B son complementadas.

Si se realiza la resta con complemento a uno, los bits del sustraendo B han de ser complementados cambiando ceros por unos y unos por ceros, lo que equivale a conectar puertas inversoras NO a la entrada de dichos terminales. El complemento a uno tiene en cuenta el acarreo final, convirtiéndose en acarreo previo de la suma por lo que la salida y entrada de acarreo serán conectadas, obteniéndose el circuito de resta binaria con sumadores que se describe en la figura 8.17. Este mismo circuito puede realizar la resta en complemento a dos eliminando la realimentación del acarreo final e introduciendo un "1" en la entrada de acarreo previo del primer sumador (el menos significativo).

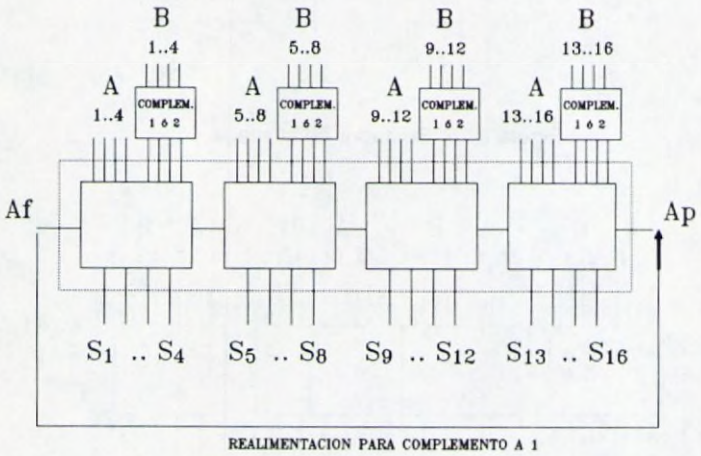


Figura 8.17. Circuito de resta binaria

*## **Ejemplo 8.41**: Analizar las operaciones que realiza el circuito de la figura 8.18 para valores de x=0 y x=1.*

Solución:

a) x=0

Las salidas de las puertas XOR son iguales que las entradas C_1 y C_2. Por tanto este circuito aritmético realiza la suma de B y C.

b) x=1

Las salidas de las puertas XOR son complementarias a las entradas C_1 y C_2. Al mismo tiempo existe un "1" en la entrada de acarreo previo. Por tanto se realiza la resta B-C en complemento a 2.

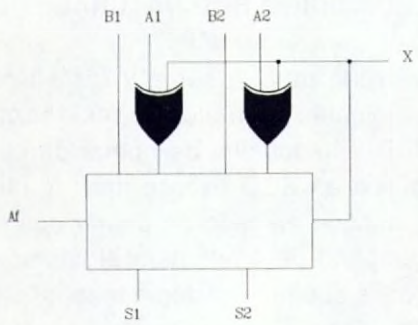


Figura 8.18

*## **Ejemplo 8.42**: Analizar las operaciones que realiza el circuito de la figura 8.19 para valores de x=0 y x=1.*

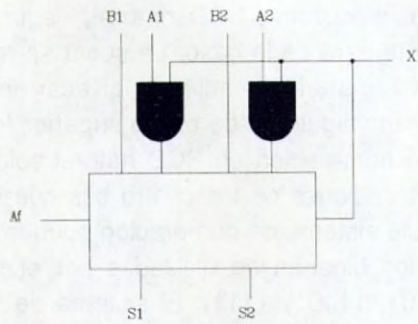


Figura 8.19

Solución:

a) x=0

Las salidas de las puertas AND son siempre "0". Por tanto este circuito aritmético realiza la suma de B y "0" o lo que es lo mismo, se transmite B a la salida.

b) x=1

Las salidas de las puertas AND son iguales que las entradas entradas C_1 y C_2. Al mismo tiempo existe un "1" en la entrada de acarreo previo. Por tanto se realiza el incremento de la suma de B y C (B+C+1).

8.7. ARITMETICA EN EL CODIGO BCD NATURAL

Igual que se ha realizado para la suma y resta binaria, se describen a continuación los procedimientos generales de realización de las operaciones de suma y resta en BCD. El esquema de operación es genérico y sirve de introducción a la aritmética en BCD exceso tres. Al utilizar bit de signo se empleará un único bit, aunque en caso de operar con registros estándar se reservará un bloque completo de 4 bits para el mismo. En este sistema se dan las mismas situaciones que en el sistema binario, produciéndose desbordamiento en los mismos casos.

8.7.1. Suma en BCD Natural

Para realizar la suma en el código de numeración BCD natural será necesario transformar los números que se estén empleando en cualquier otro código de numeración, al sistema BCD natural, según los procedimientos analizados anteriormente para cada caso. La suma se realiza para cada uno de los términos en BCD natural, añadiendo el acarreo resultante en cada bloque de suma al término siguiente de orden superior (más significativo).

Como el sistema de numeración en BCD natural sólo emplea diez de las dieciséis posibles combinaciones de los cuatro bits que lo constituyen, al sumar dos términos de este sistema de numeración pueden aparecer algunas de esas seis combinaciones binarias no utilizadas por el sistema BCD natural (1010, 1011, 1100, 1101, 1110 y 1111). El sistema de corrección de dichas combinaciones no utilizadas o prohibidas, se basa en los siguientes apartados:

* Si los resultados de las sumas parciales son **SUPERIORES A "1001"**, se corrige **AÑADIENDO "0110"**, equivalente al número 6 en decimal.

* Si los resultados de las sumas parciales son **INFERIORES O IGUALES A "1001" NO ES NECESARIO REALIZAR CORRECCION**.

Tomando como base el sistema decimal para una más fácil comprensión, se pueden analizar las reglas de corrección del sistema en BCD natural. Los términos de este sistema ocupan las combinaciones de cuatro bits equivalentes a los números decimales del 0 al 9, quedando sin utilizar del 10 al 15. Cuando se indica que un término en BCD natural es superior al "1001", quiere decir que ocupa una de esas posiciones decimales que se pueden denominar como prohibidas. Si se indica que es inferior o igual al "1001", ocupa una de las posiciones válidas del sistema de numeración.

Un número prohibido del 10 al 15 en decimal, puede pasar a una combinación permitida añadiendo 6 posiciones, o lo que es lo mismo, añadiendo "0110". Si al realizar la suma de elementos y posibles acarreos en el término más significativo, el de la izquierda, se produjera un acarreo final, se sumará a un nuevo término que se crea añadiendo el valor "0000", equivalente al cero en decimal, a la izquierda del término en el que operaba, convirtiéndose ahora este nuevo término en el mas significativo de la expresión final.

En caso de que alguno de los términos de la suma posea menor número de elementos que el otro, se deberá proceder a la igualación de los mismos mediante la introducción de elementos "0000", de tal manera que siempre se sumen un número equivalente de elementos. Si al añadir los acarreos iniciales a los términos más significativos, alguno de los términos que antes no superaban el valor "1001", ahora sí lo superan, será necesario convertir estos términos al valor correcto, añadiéndole el término "0110" de corrección, produciéndose por tanto una segunda operación de conversión. Conviene tener muy en cuenta que en esta segunda fase de corrección se añadirá "0110" únicamente a las nuevas sumas parciales que superaban el valor "1001". Como resumen, se puede indicar que todos los acarreos se deben utilizar no importando el momento en que se obtengan y que se convertirán todos los términos superiores a "1001" mediante la adición del número binario "0110". El proceso de suma de acarreos se puede realizar a medida que aparezcan, o esperar a operaciones más avanzadas donde se trasladen todos al mismo tiempo. Sólo se pueden convertir los términos superiores a "1001" una sola vez dados los límites máximos del sistema de numeración.

*## **Ejemplo 8.43**: Realizar la suma en BCD Natural de los números decimales 1847 y 835.*

Solución:

	0001	1000	0100	0111	
+	0000	1000	0011	0101	
	0001	10000	0111	1100	
+		0110		0110	Corrección
	0001	0110	0111	0010	
+	1		1		Acarreo
	0010	0110	1000	0010	

*## **Ejemplo 8.44**: Realizar la suma en BCD Natural de los números decimales 3974 y 6835.*

Solución:

	0011	1001	0111	0100	
+	0110	1000	0011	0101	
	1001	0001	1010	1001	
+	1				Acarreo
	1010	0001	1010	1001	
+	0110	0110	0110		Corrección
	0000	0000	0111	0000	1001
+	1		1		Acarreo
	0001	0000	1000	0000	1001

8.7.2. **Resta en BCD Natural**

La operación de resta se efectua de manera equivalente a la realizada para el sistema binario y con las indicaciones particulares indicadas para la suma en BCD natural. La resta se realiza mediante la suma del minuendo y el complemento a nueve del sustraendo (En este caso, al no utilizarse las 16 combinaciones binarias de los 4 bits, el complemento se calcula respecto al máximo valor que puede tomar este sistema de numeración, concretamente el 9 en decimal ó "1001" en binario). Por tanto, todas las condiciones que se detallaron en el apartado anterior para la suma, serán de aplicación para la resta, salvo determinadas particularidades en los acarreos que se analizarán más adelante.

Las sumas parciales se realizarán transmitiendo los acarreos parciales a los bloques más significativos. excepto **SI EXISTE ACARREO FINAL**, que **SE SUMARA AL BIT MENOS SIGNIFICATIVO**. El sistema de corrección de las combinaciones no utilizadas o prohibidas, se realiza de idéntica forma que para la suma en BCD natural. En caso de que alguno de los términos de la resta posea menor número de elementos que el otro, se deberá proceder a la igualación de los mismos mediante la introducción de elementos "0000", de tal manera que siempre se sumen un número equivalente de elementos.

La suma es **POSITIVA SI SE PRODUCE ACARREO FINAL**, mientras que será **NEGATIVA SI NO SE PRODUCE ACARREO FINAL**, estando en este caso el resultado expresado en complemento a 9, y por tanto, es

necesario descomplementarlo para obtener la resta final. Si se utiliza bit de signo, inicialmente los dos términos llevarán el valor "0" en el bit de signo, considerando como negativo el signo de la operación. Al efectuar el complemento a nueve del sustraendo, la resta pasará a ser una suma, por lo que el sustraendo se convertirá en un número negativo, y el bit de signo del mismo pasará a ser un "1". El acarreo del último término de la suma se añadirá al bit de signo, realizándose una nueva operación de suma. Si aparece acarreo final en esta suma, se añadirá de acuerdo con la regla anterior al bit menos significativo, el de la derecha.

Ejemplo 8.45: *Realizar la resta en BCD Natural de los números decimales 6835 y 974.*

Solución:

	0	0110	1000	0011	0101	M
+	1	1001	0000	0010	0101	Cto. 9
	1	1111	1000	0101	1010	
+	0110				0110	Corrección
	1	0101	1000	0101	0000	
+	1			1		Acarreo
	0	0101	1000	0110	0000	
+					1	Acarreo
	0	0101	1000	0110	0001	Resta final

Ejemplo 8.46: *Realizar la resta en BCD Natural de los números decimales 974 y 6835.*

Solución:

	0	0000	1001	0111	0100	M
+	1	0011	0001	0110	0100	Cto. 9
	1	0011	1010	1101	1000	
+			0110	0110		Corrección
	1	0011	0000	0011	1000	
+		1	1			Acarreo
	1	0100	0001	0011	1000	Resta en cto. 9
	1	0101	1000	0110	0001	Resta final

Ejemplo 8.47: *Realizar la resta en BCD Natural de los números decimales 8357 y 38.*

Solución:

	0	1000	0011	0101	0111	M
+	1	1001	1001	0110	0001	Cto. 9
	1	10001	1100	1011	1000	
+		0110	0110	0110		Corrección
	1	0111	0010	0001	1000	
+	1	1	1			Acarreo
	0	1000	0011	0001	1000	
+					1	Acarreo
	0	1000	0011	0001	1001	Resta final

Ejemplo 8.48: *Realizar la resta en BCD Natural de los números decimales 38 y 8357.*

Solución:

	0	0000	0000	0011	1000	M
+	1	0001	0110	0100	0010	Cto. 9
	1	0001	0110	0111	1010	
+					0110	Corrección
	1	0001	0110	0111	0000	
+				1		Acarreo
	1	0001	0110	1000	0000	Resta en cto. 9
	1	1000	0011	0001	1001	Resta final

Ejemplo 8.49: *Realizar la resta en BCD Natural de los números decimales -6835 y +974.*

Solución:

	1	0011	0001	0110	0100	Cto. 9 de M
+	1	1001	0000	0010	0101	Cto. 9 de S
	10	1100	0001	1000	1001	
+		0110				Corrección
	0	0010	0001	1000	1001	
+	1				1	Acarreo
	1	0010	0001	1000	1010	
+					0110	Corrección
	1	0010	0001	1000	0000	
+				1		Acarreo
	1	0010	0001	1001	0000	Resta en cto. 9
	1	0111	1000	0000	1001	Resta final

*## **Ejemplo 8.50**: Realizar la resta en BCD Natural de los números decimales -3974 y +6835.*

Solución:

	1	0110	0000	0010	0011	Cto. 9 de M
+	1	0011	0001	0110	0100	Cto. 9 de S
	0	1001	0001	1000	0111	
+					1	Acarreo
	0	1001	0001	1000	1000	Desbordamiento

*## **Ejemplo 8.51**: Realizar la resta en BCD Natural de los números decimales -6835 y -974.*

Solución:

	1	0011	0001	0110	0100	Cto. 9 de M
+	0	0000	1001	0111	0100	Cto. 9 de cto. 9 de S
	1	0011	1010	1101	1000	
+			0110	0110		Corrección
	1	0011	0000	0011	1000	
+		1	1			Acarreo
	1	0100	0001	0011	1000	Resta en cto. 9
	1	0101	1000	0110	0001	Resta final

*## **Ejemplo 8.52**: Realizar la resta en BCD Natural de los números decimales 38 y -8357.*

Solución:

	0	0000	0000	0011	1000	M
+	0	1000	0011	0101	0111	Cto. 9 de cto.9 de S
	0	1000	0011	1000	1111	
+					0110	Corrección
	0	1000	0011	1000	0101	
+				1		Acarreo
	0	1000	0011	1001	0101	Resta final

8.8. CIRCUITO DE SUMA EN BCD NATURAL

El sistema BCD está representado por las diez primeras combinaciones (del 0 al 9) de cuatro dígitos binarios. Si se suman dos números en BCD se puede obtener un valor máximo de 18 en decimal ó "10010" en binario. Si a estos dos números se le añade un tercer bit de acarreo previo tal como se realiza en el sumador binario, se alcanza como valor máximo el 19 en decimal ó "10011" en binario. Ambos números exceden el valor máximo permitido del sistema BCD, por lo que todas las cifras que excedan el valor "1001" han de ser corregidas añadiéndole "0110".

Por tanto el esquema inicial del sumador BCD puede suponerse compuesto por una primera etapa de suma binaria de los valores BCD a sumar y por una segunda etapa de suma binaria donde las entradas serán la suma de la primera etapa binaria y la corrección "0110" para los resultados no comprendidos en el código BCD. Para diseñar el circuito se realiza el análisis del acarreo existente en la primera etapa de suma de los términos BCD:

* Si la combinación resultante en la salida del primer sumador es permitida en el sistema BCD, se considera correcto el resultado y por tanto el acarreo final será "0".
* Si la combinación mencionada no es BCD, se corrige añadiéndole "0110" con lo cuál siempre existirá acarreo final ("1"), que además tendrá que ser transmitido a la siguiente etapa sumadora.

Teniendo en cuenta estos datos, se rellena la tabla de Karnaugh correspondiente a la función acarreo final, con valores "0" para las diez primeras combinaciones, "1" para las diez siguientes hasta llegar al valor máximo 19, y "-" para las restantes combinaciones no especificadas, obteniendo los resultados de la tabla 8.5.

$S_3 S_4$ \ $A_f S_1 S_2$	000	001	011	010	110	111	101	100
00	0	0	1	0	-	-	-	1
01	0	0	1	0	-	-	-	1
11	0	0	1	1	-	-	-	1
10	0	0	1	1	-	-	-	1

Tabla 8.5. Diseño del sumador BCD natural

$$F = A_f + S_1 S_2 + S_1 S_3 \qquad [8.11]$$

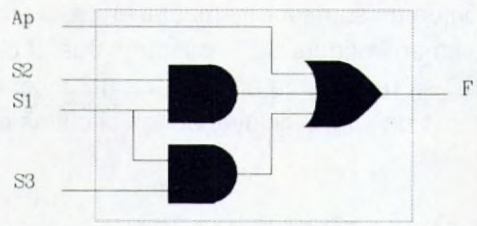


Figura 8.20. Función acarreo

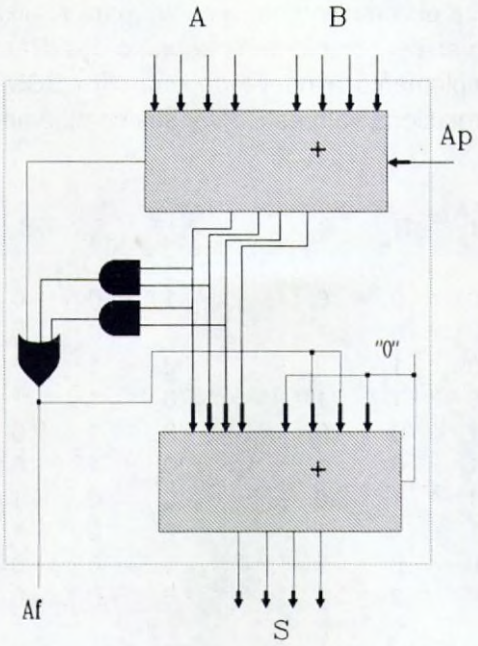


Figura 8.21. Sumador elemental en BCD natural

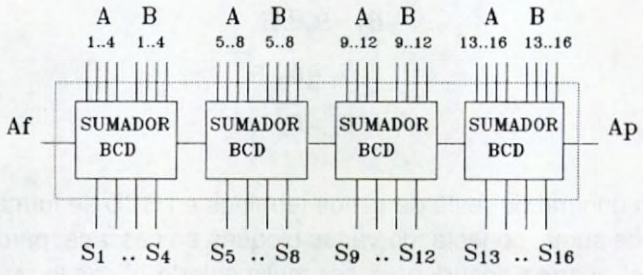


Figura 8.22. Sumador BCD de 4 cifras

La representación de un sumador elemental en BCD empleando sumadores binarios se representa en la figura 8.21, mientras que el circuito de suma de 4 cifras en código BCD se representa en la figura 8.22, conectando en cascada las etapas elementales de suma incluyendo los circuitos de corrección.

8.9. CIRCUITO DE RESTA EN BCD NATURAL

Siguiendo la línea descrita para las operaciones binarias, el restador en BCD se realizará a través del sumador del mismo sistema. Sin embargo este tipo de código utiliza el complemento a nueve para realizar la transformación del sustraendo en vez del complemento a uno o dos. Para estudiar el circuito que realiza el complemento a nueve de una cifra BCD se construye la siguiente tabla que relaciona valores BCD y sus complementos a nueve:

B_1	B_2	B_3	B_4	B_1^1	B_2^1	B_3^1	B_4^1
0	0	0	0	1	0	0	1
0	0	0	1	1	0	0	0
0	0	1	0	0	1	1	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	0	0
0	1	1	0	0	0	1	1
0	1	1	1	0	0	1	0
1	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0

Tabla 8.6. Complemento a 9

$$B_1^1 = \overline{B}_1 \overline{B}_2 \overline{B}_3 \quad [8.12]$$

$$B_2^1 = B_2 \oplus B_3 \quad [8.13]$$

$$B_3^1 = B_3 \quad [8.14]$$

$$B_4^1 = \overline{B}_4 \quad [8.15]$$

El circuito general de resta de varios términos en BCD se formará de manera similar al de suma, conectando varios bloques en cascada, pero teniendo en cuenta que el acarreo final deberá ser realimentado al comienzo de la suma, conectándolo al acarreo previo. Dependiendo del bit de signo o del acarreo

final, el resultado será positivo o negativo, viniendo expresado en BCD o en su complemento a nueve, necesitando en este último caso una posterior conversion a BCD mediante nueva complementación. De esta forma se pueden obtener circuitos como el de la figura 8.24 que resta términos de 4 cifras en BCD.

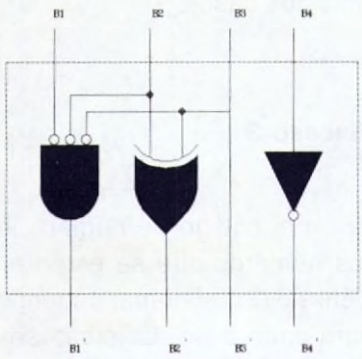


Figura 8.23. Circuito de complemento a 9

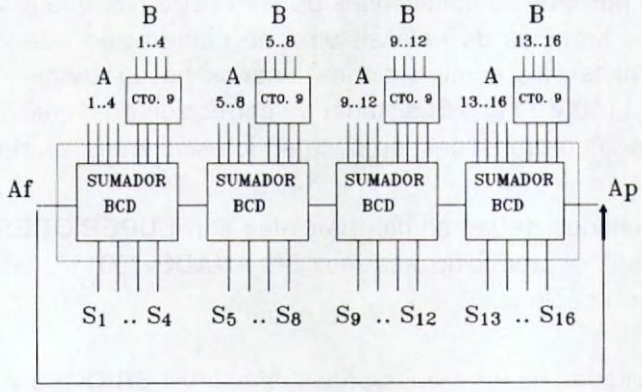


Figura 8.24. Restador en BCD natural

8.10. **ARITMETICA EN EL CODIGO BCD EXCESO-3**

Las reglas que se emplean en este sistema de numeración son similares a las del código BCD natural, existiendo diferencia únicamente en los elementos empleados para realizar las conversiones. La importancia de este código de numeración, reside principalmente en su caracter de

autocomplementario, lo que repercutirá en una notable simplicidad de alguunas operaciones que se realicen con el mismo. Al utilizar bit de signo se empleará un único bit, aunque en caso de operar con registros estándar se reservará un bloque completo de 4 bits para el mismo. En este sistema se dan las mismas situaciones que en el sistema binario, produciéndose desbordamiento en los mismos casos.

8.10.1. **Suma en BCD Exceso-3**

Para realizar la suma en el código de numeración BCD Exceso-3 será necesario transformar los números que se estén empleando en cualquier otro código de numeración a dicho sistema, según los procedimientos analizados anteriormente para cada caso. La suma se realiza para cada uno de los términos en BCD Exceso-3, añadiendo el acarreo resultante en cada bloque de suma al término siguiente de orden superior (más significativo).

Como el sistema de numeración en BCD Exceso-3 sólo emplea diez de las dieciséis posibles combinaciones de los cuatro bits que lo constituyen, al sumar dos términos de este sistema de numeración pueden aparecer algunas de esas seis combinaciones binarias no utilizadas (0000, 0001, 0010, 1101, 1110 y 1111). El sistema de corrección de dichas combinaciones no utilizadas o prohibidas, se basa en los siguientes apartados:

* Si los resultados de las sumas parciales son **SUPERIORES A "1111"**, es decir, se ha producido acarreo, **SE AÑADE "0011"**, equivalente al número 3 en decimal.

* Si los resultados de las sumas parciales son **INFERIORES A "1111"**, es decir, no se ha producido acarreo, **SE RESTA "0011"**, equivalente al número 3 en decimal, ó para más comodidad, se añade el número "1101" (complemento a 2 del "0011"), equivalente al número 13 en decimal.

Basándonos en términos del sistema decimal para una más fácil comprensión, se pueden analizar las reglas de corrección del sistema en BCD Exceso-3. Los términos de este sistema ocupan las combinaciones de cuatro bits equivalentes a los números decimales del 2 al 12, quedando sin utilizar del 0 al 2 y del 13 al 15. Cuando se indica que un término en BCD Exceso-3 es superior al "1111" ó tiene acarreo, quiere decir que ocupa una

de las posiciones decimales que se pueden denominar como prohibidas, del 0 al 2. Si se indica que es inferior al "1111", puede ocupar una de las posiciones decimales prohibidas del 13 al 15. Los números decimales del 0 al 2 pueden pasar a una combinación permitida añadiendo 3 posiciones, o lo que es lo mismo, sumándoles "0011". Los números 13 al 15 en decimal pueden pasar a una combinación permitida restando 3 posiciones o su equivalente binario "0011" (sumando "1101" si se utiliza complemento a 2).

Al realizar la corrección de términos, se ha hecho mención a las combinaciones prohibidas, pero en este sistema de numeración se deben analizar también las combinaciones permitidas, ya que no se comportan como otros sistemas. Veamos por ejemplo el caso del cero, representado por "0011". Si sumamos dos términos cero en decimal, el resultado será cero, pero si realizamos la suma en BCD Exceso-3, el resultado será "0110", correspondiente al término 3 y no al término 0. Esto quiere decir, que existe un exceso de tres unidades que es necesario corregir, restándolas o como se indicó anteriormente, añadiendo "1101".

De esta manera, todos los términos deben ser compensados, y la regla definitiva puede expresarse indicando que **TODOS LOS TERMINOS SERAN CORREGIDOS. SI EXISTE ACARREO SE AÑADE 0011 Y SI NO EXISTE ACARREO SE AÑADE 1101. LOS ACARREOS OBTENIDOS EN LA CORRECCION SERAN DESPRECIADOS.** Esta indicación requiere que los acarreos producidos en cualquier operación que no corresponda con la corrección, debe ser añadido antes de realizar dicha corrección para evitar errores.

Si al sumar dos números en BCD Exceso-3, se pasa a un sistema equivalente a BCD Exceso-6, al corregir de esta forma el defecto se obtienen resultados en BCD Exceso-3. Si en el término más significativo, el de la izquierda, se produce acarreo final, se sumará a un nuevo término, el "0011", equivalente del cero decimal en BCD Exceso-3, constituyendo el término más significativo de la expresión final. Es importante hacer notar que si se crea un nuevo término añadiendo "0011", no debe ser corregido, pero si se añaden dos términos "0011" en minuendo y sustraendo, entonces sí se hace necesaria la corrección. Como siempre, en caso de que alguno de los términos de la suma posea menor número de elementos que el otro, se deberá proceder a la igualación de los mismos mediante la introducción de elementos "0011", de forma que se sumen un número equivalente de elementos.

*## **Ejemplo 8.53**: Realizar la suma en BCD Exceso-3 de los números decimales 641 y 502.*

Solución:

		1001	0111	0100	
+		1000	0011	0101	
		10001	1010	1001	
+		0011	1101	1101	Corrección
	0011	0100	0111	0110	
+	1				Acarreo
	0100	0100	0111	0110	

*## **Ejemplo 8.54**: Realizar la suma en BCD Exceso-3 de los números decimales 641 y 5504.*

Solución:

	0011	1001	0111	0100	
+	1000	1000	0011	0111	
	1011	10001	1010	1011	
+	1101	0011	1101	1101	Corrección
	1000	0100	0111	1000	
+	1				Acarreo
	1001	0100	0111	1000	

8.10.2. **Resta en BCD Exceso-3**

Para realizar la resta en el código de numeración BCD Exceso-3 será necesario, al igual que se indicó para la operación de suma, transformar los números que se estén empleando en cualquier otro código de numeración. Al igual que en la resta en BCD natural, la resta en BCD Exceso-3 se realiza mediante la suma del minuendo y el complemento del sustraendo. Al tener este sistema 10 de las 16 posibles combinaciones binarias, el complemento se deberá realizar a 9 igual que en el sistema BCD natural. Sim embargo, al calcular el complemento es importante notar que se realiza por dos veces un aumento de tres unidades en el cálculo operativo, convirtiéndose aparente-

mente en el complemento a quince. El complemento a nueve en este sistema de un número cualquiera se obtiene mediante la diferencia entre 9 y dicho número (Por ej. el cto. a 9 del número 2 es el 7, pero en BCD Exceso-3 el 2 es "0101" y el 7 es "1010"). En el sistema de numeración BCD Exceso-3, al ser un sistema autocomplementario, el complemento a 9 se obtiene cambiando los ceros por unos y los unos por ceros, es decir, hallando el complemento de cada uno de los dígitos que lo constituyen.

El acarreo que se produzca en cada suma parcial, se añade al término contiguo más significativo. **SI SE PRODUCE ACARREO FINAL SE AÑADIRA AL BIT MENOS SIGNIFICATIVO**.

Como en todas ls operaciones de resta descritas, **SI SE PRODUCE ACARREO FINAL EL RESULTADO ES POSITIVO. SI NO SE PRODUCE ACARREO FINAL EL RESULTADO ES NEGATIVO Y ESTA EN COMPLEMENTO A 9**. Es por tanto necesario descomplementar dicho resultado para obtener la resta final.

Si se utiliza bit de signo, al efectuar el complemento a nueve del sustraendo, la resta pasará a ser una suma, y el bit de signo del mismo será un "1". El acarreo del último término de la suma se añadirá al bit de signo. Si aparece acarreo final en esta suma, se añadirá de acuerdo con la regla anterior al bit menos significativo, el de la derecha. Todas las condiciones de la suma son de aplicación a la operación de resta, especialmente el sistema de corrección de combinaciones no utilizadas o prohibidas y de combinaciones permitidas.

*## **Ejemplo 8.55**: Realizar la resta en BCD Exceso-3 de los números decimales 3502 y 641.*

Solución:

	0110	1000	0011	0101	M
+	1100	0110	1000	1011	Cto. 9
	0010	1110	1011	0000	
+			1	1	Acarreo
	0010	1110	1100	0001	
+	0011	1101	1101	0011	Corrección
	0101	1011	1001	0100	Resta final

Ejemplo 8.56: *Realizar la resta en BCD Exceso-3 de los números decimales 641 y 3502.*

Solución:

	0011	1001	0111	0100	M
+	1001	0111	1100	1010	Cto. 9
	1100	0000	0011	1110	
+	1	1			Acarreo
	1101	0001	0011	1110	
+	1101	0011	0011	1101	Corrección
	1010	0100	0110	1011	Resta en cto. 9
	0101	1011	1001	0100	Resta final

Ejemplo 8.57: *Realizar la resta en BCD Exceso-3 de los números decimales -3502 y +641.*

Solución:

	1	1001	0111	1100	1010	Cto. 9 de M
+	1	1100	0110	1000	1011	Cto. 9 de S
	10	10101	1101	10100	10101	
+		0011	1101	0011	0011	Corrección
	0	1000	1010	0111	1000	
+	1		1	1	1	Acarreo
	1	1000	1011	1000	1001	Resta en cto. 9
	1	0111	0100	0111	0110	Resta final

Ejemplo 8.58: *Realizar la resta en BCD Exceso-3 de los números decimales -3502 y +7641.*

Solución:

	1	1001	0111	1100	1010	Cto. 9 de M
+	1	0101	0110	1000	1011	Cto. 9 de S
	10	1110	1101	10100	10101	
+		0011	1101	0011	0011	Corrección
	0	0001	1010	0111	1000	
+			1	1	1	Acarreo
	0	0001	1011	1000	1001	Desbordamiento

*## **Ejemplo 8.59**: Realizar la resta en BCD Exceso-3 de los números decimales -3502 y -641.*

Solución:

	1	1001	0111	1100	1010	Cto. 9 de M
+	0	0011	1001	0111	0100	Cto. 9 de cto. 9 de S
	1	1100	10000	10011	1110	
+		1101	0011	0011	1101	Corrección
	1	1001	0011	0110	1011	
+		1	1			Acarreo
	1	1010	0100	0110	1011	Resta en cto. 9
	1	0101	1011	1001	0100	Resta final

*## **Ejemplo 8.60**: Realizar la resta en BCD Exceso-3 de los números decimales 3502 y -641.*

Solución:

	0	0110	1000	0011	0101	M
+	0	0011	1001	0111	0100	Cto. 9 de cto. 9 de S
	0	1001	10001	1010	1001	
+		1101	0011	1101	1101	Corrección
	0	0110	0100	0111	0110	
+		1				Acarreo
	0	0111	0100	0111	0110	Resta final

8.11. CIRCUITO DE SUMA EN BCD EXCESO-3

El sistema BCD Exceso-3 está representado por diez combinaciones de 4 dígitos binarios, equivalentes a los números 3 a 12 del sistema decimal y al igual que el sistema BCD natural tiene seis combinaciones no empleadas de las dieciséis posibles que formarían los cuatro bits que lo constituyen. Al sumar dos números en BCD Exceso-3 se puede obtener un valor máximo de 24 en decimal ó "11000" en binario si no se tiene en cuenta el acarreo previo, ó 25 en decimal y "11001" en binario si se tiene en cuenta el acarreo.

Como el sistema BCD Exceso-3 tiene una forma peculiar de corrección, añadiendo "0011" a los términos superiores a "1111" o términos con acarreo

final, y añadiendo "1101" a los términos iguales o inferiores a "1111", equivalentes a términos sin acarreo, el acarreo de cada etapa de suma será directo y no necesita ningún circuito combinacional para su obtención como puede además comprobarse de los valores procedentes de la tabla de Karnaugh, que analizan la función acarreo, igual que se analizó en BCD natural.

$S_3 S_4$ \ $A_f S_1 S_2$	000	001	011	010	110	111	101	100
00	0	0	0	0	1	-	1	1
01	0	0	0	0	1	-	1	1
11	0	0	0	0	-	-	1	1
10	0	0	0	0	-	-	1	1

Tabla 8.7. Diseño del sumador BCD Exceso-3

$$F=A_f \qquad [8.16]$$

El esquema del sumador BCD Exceso-3 estará compuesto por una primera etapa de suma binaria de los valores de entrada y por una segunda etapa de suma binaria donde las entradas serán la suma de la primera etapa binaria y la corrección correspondiente. Si el resultado tiene acarreo se suma "0011" y si no lo tiene se suma "1101".

		A	B	C	D
Acarreo = 1	+	0	0	1	1
Acarreo = 0	+	1	1	0	1

Se observa que los bits A y B se corresponden con el complemento del acarreo, que el bit C es igual que el acarreo y que el bit D siempre será "1", por lo que a partir del bit de acarreo final se puede formar el circuito completo de suma en BCD Exceso-3. Para sumar varios bits se realizará conectando en cascada tantas etapas sumadoras elementales como sean necesarias. En la figura 8.25 se representa el esquema del sumador elemental en código BCD Exceso-3 y en la figura 8.26 un montaje en cascada para sumar 4 cifras.

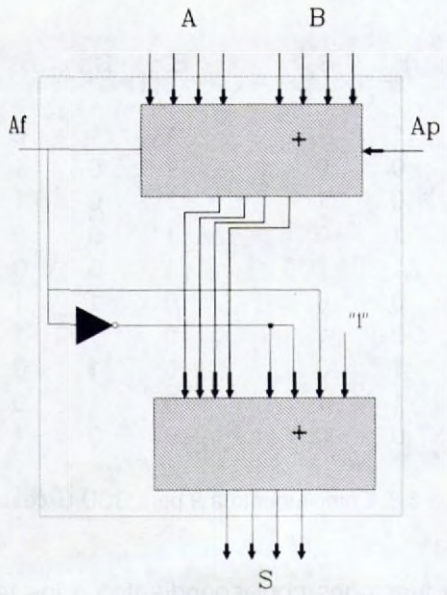


Figura 8.25. Circuito de suma elemental en BCD Exceso-3

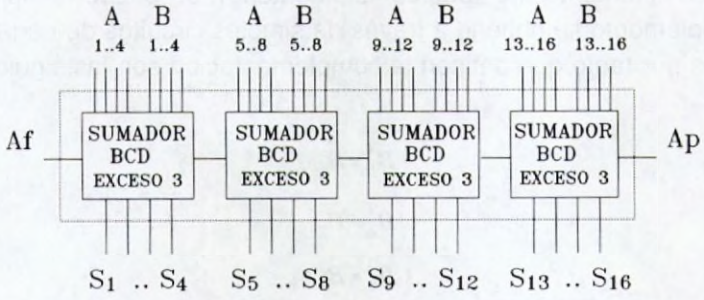


Figura 8.26. Sumador de 4 cifras BCD Exceso-3

8.12. **CIRCUITO DE RESTA EN BCD EXCESO-3**

Siguiendo la misma línea que la descrita para otros sistemas, el restador BCD Exceso-3 se realiza a través del sumador del mismo sistema. Este tipo de circuito utiliza el complemento a 9 para realizar la transformación del sustraendo. Para estudiar el circuito que realiza el complemento a 9 de una cifra BCD Exceso-3 se construye la siguiente tabla 8.8 que relaciona valores del sistema y sus complementos.

B_1	B_2	B_3	B_4	B_1^1	B_2^1	B_3^1	B_4^1
0	0	1	1	1	1	0	0
0	1	0	0	1	0	1	1
0	1	0	1	1	0	1	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	0	0
1	0	0	0	0	1	1	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	0	1
1	0	1	1	0	1	0	0
1	1	0	0	0	0	1	1

Tabla 8.8. Complemento a 9 para BCD Exceso-3

Si se estudian las funciones correspondientes a los términos B_1, B_2, B_3 y B_4 y sus complementos, se obtendrían las funciones a través de las tablas de simplificación de Karnaugh. Sin embargo, observando la tabla 8.8 y el concepto de complemento a 9 (autocomplementario), se puede comprobar que dicho complemento se obtiene a través de simples circuitos de negación. Las ecuaciones por tanto que definen la complementación son las siguientes:

$$B_1^1 = \overline{B}_1 \qquad [8.17]$$

$$B_2^1 = \overline{B}_2 \qquad [8.18]$$

$$B_3^1 = \overline{B}_3 \qquad [8.19]$$

$$B_4^1 = \overline{B}_4 \qquad [8.20]$$

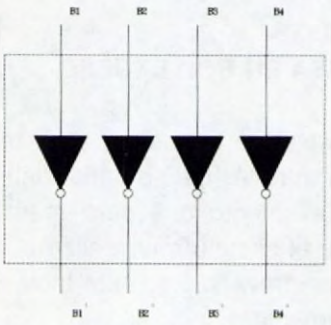


Figura 8.27. Circuito de complemento a 9 para BCD Exceso-3

El circuito general de resta de varios términos se formará de manera similar al realizado para BCD natural, conectando varios bloques en cascada, pero teniendo en cuenta que el acarreo final es realimentado al comienzo de la suma, conectándolo con el acarreo previo. De esta forma se pueden obtener circuitos como el de la figura 8.28 que resta términos de 4 cifras en BCD Exceso-3.

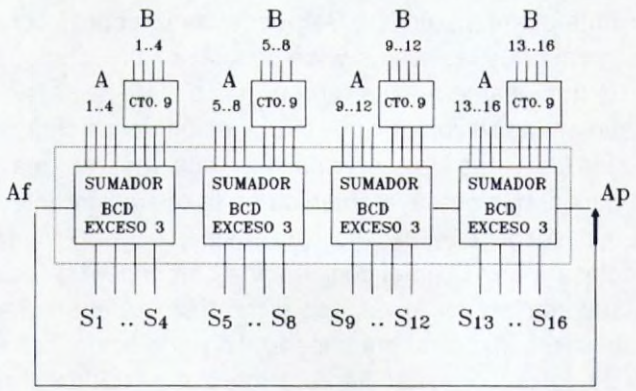


Figura 8.28. Circuito de resta en BCD Exceso-3

8.13. **MULTIPLICACION BINARIA**

Se realiza exactamente igual que la estructura del sistema decimal, debido a que los coeficientes empleados tanto para el multiplicador como para el multiplicando son el "0" y el "1" binarios. Esta operación se denomina de suma-desplazamiento, dado que en caso de multiplicar por un "1" se sumará el multiplicando, y en caso de multiplicar por un "0" se desplazará el siguiente valor a sumar, que puede ser nuevamente el multiplicando si aparece otro "1" en el multiplicador.

*## **Ejemplo 8.61**: Realizar la multiplicación binaria de los números decimales 118 y 27.*

Solución:

```
        1110110          *      118
*       11011                   27
  -------------          ---------
        1110110                826
       1110110                 236
      1110110                  ----
     1110110                   3186
  -------------
  110001110010
```

8.14. **DIVISION BINARIA**

Se realiza exactamente igual que en el sistema decimal, siendo denominada de resta- desplazamiento, dado que al ser los coeficientes empleados el "0" y el "1" binarios, se restará el divisor en los casos en que se obtenga un "1" en el cociente y se desplazará dicho divisor para ser restado en caso de que en el cociente aparezca un "0". Se puede comprobar el correcto cumplimiento de las normas de las restas y los desplazamientos según los valores del cociente. Es interesante resaltar que en el caso de la división binaria, es más sencillo que en la división decimal la obtención de las cifras del cociente.

Antes de fijar las cifras del cociente, hay que realizar una resta previa entre el resto parcial que se haya obtenido en la operación anterior y el valor del divisor. En caso de que el valor del resto parcial sea superior al del cociente, se colocará un "1" en el cociente y se procederá a realizar la resta, y en caso de que sea inferior el valor de dicho resto se añadirá el número de bits necesarios para que dicha cifra supere al divisor, colocando en el cociente el número de ceros correspondiente a los bits añadidos al resto parcial. Los bits que se añaden son los correspondientes al dividendo, añadiéndose ceros en caso de que se agoten todas las cifras del dividendo. Este proceso se realiza desde la obtención del primer bit del cociente.

*## **Ejemplo 8.62**: Realizar la división binaria de los números decimales 118 y 27.*

Solución:

```
1 1 1 0 1 1 0 | 1 1 0 1 1
-------------
1 1 0 1 1     | 1 0 0 , 0 1 0 1 1 1 1 0 1
-------------
0 0 0 1 0 1 0 0 0
    1 1 0 1 1
  -------------
  0 0 1 1 0 1 0 0
      1 1 0 1 1
    -------------
    1 1 0 0 1 0
      1 1 0 1 1
    -------------
    0 1 0 1 1 1 0
        1 1 0 1 1
      -------------
      0 1 0 0 1 1 0
          1 1 0 1 1
        -------------
        0 0 1 0 1 1 0 0
            1 1 0 1 1
          -------------
          0 1 0 0 0 1
```

8.15. OPERADORES - LA UNIDAD ARITMETICA LOGICA (ALU)

Un circuito digital fundamental en la concepción de los equipos de proceso de datos es el que se se encarga de realizar las operaciones aritméticas necesarias para procesar información, así como para realizar las distintas funciones lógicas de acuerdo con lo expresado en temas anteriores. Este circuito digital es un compendio de pequeños circuitos digitales, conjuntados en un único bloque integrado de enormes posibilidades denominado unidad aritmética lógica o ALU.

La ALU estándar realiza 32 funciones diferentes, correspondiendo 16 de ellas a operaciones aritméticas y otras 16 a operaciones lógicas. Un modelo muy utilizado es el SN74181. La constitución y el estudio de la ALU se realiza a través del estudio de los elementos denominados operadores. Son circuitos aritméticos que tienen además la posibilidad de ser utilizados en modo lógico. Su esquema simple responde al diagrama representado en la figura 8.29.

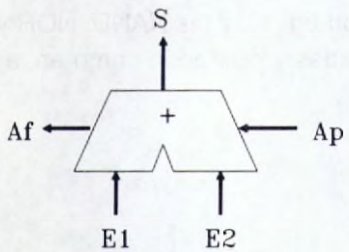


Figura 8.29. Operador de suma

$$R=E_2+E_1+A_p \qquad [8.21]$$

Este circuito realizará la suma aritmética de tres datos de entrada, suministrando un resultado equivalente al del sumador completo. Sin embargo, no es esta la única opción de operación que puede realizar, ya que si se complementan una de las entradas de datos y la entrada de acarreo previo, se puede conseguir realizar la operación de resta binaria, quedando los resultados según se indica en la figura 8.30. (La salida P_i estará complementada para hacer equivalentes los resultados globales de resta y suma complementada)

$$R=E_2+\overline{E}_1+\overline{P}_p \qquad [8.22]$$

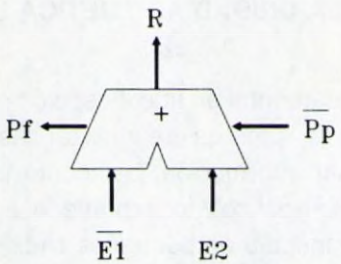


Figura 8.30. Operador de resta

Si el operador de la figura 8.29 se analiza en forma lógica aunque aparentemente realice operaciones aritméticas, se comprueba que la salida S es equivalente a la función lógica O-Exclusiva y la salida A_f equivale a la función lógica Producto (AND). La unión de ambas salidas proporciona la función lógica de Suma (OR). De esta forma el operador se denomina aritmético-lógico. Otras funciones como la complementación (NO) se obtiene mediante suma de las entradas con un "1", y las NAND, NOR y XNOR realimentando la salida a una de las entradas y operando como en la puerta NO.

Figura 8.31. Registro del operador

El operador necesitará de elementos auxiliares de almacenamiento temporal de los datos de entrada y salida. Estos elementos o registros, se denominan "Acumulador" y "Registro de Estado" cuando almacenan el resultado final y el acarreo final respectivamente. Cada uno de los registros de entrada y salida de datos puede operar bien en forma serie, introduciendo uno a uno los bits y desplazándolos, o en forma paralela, cargándose y descargándose todos los bits simultáneamente.

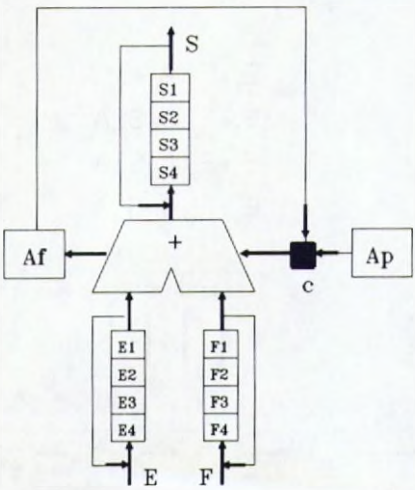


Figura 8.32. Operador serie

El operador serie obliga al funcionamiento individual para cada terna de bits de entrada. Para realizar por tanto una suma de 4 bits como en la figura 8.32, se realizarán consecutivamente 4 operaciones de suma.

Etapa 1: $E_1+F_1+A_p=S_1,A_1$

Etapa 2: $E_2+F_2+A_1=S_2,A_2$

Etapa 3: $E_3+F_3+A_2=S_3,A_3$

Etapa 4: $E_4+F_4+A_3=S_4,A_f$

Como en cada operación de suma binaria se puede producir acarreo final, que deberá ser tenido en cuenta en la siguiente etapa de suma, la salida de acarreo final estará conectada con la de acarreo de entrada a través de una puerta lógica (c) que permita su utilización y que mediante una señal de control elimine el posible acarreo precedente en la primera etapa sumadora.

El operador paralelo de la figura 8.33, procesa simultáneamente los 4 bits que se introducen en los 4 operadores elementales. La ventaja es notoria, pero es necesario resolver el inconveniente de la generación de acarreos que han de añadirse a las sucesivas etapas de operación y que aparecen una vez que se ha efectuado la suma de las 4 células. Para ello se utiliza un circuito denominado generador rápido de acarreo que procesa todos los acarreos, reduciendo considerablemente el tiempo necesario para transmitirlos en cascada.

Etapa 1:

$$E_1+F_1+A_p=S_1,A_1$$
$$E_2+F_2+0=S_2,A_2$$
$$E_3+F_3+0=S_3,A_3$$
$$E_4+F_4+0=S_4,A_f$$

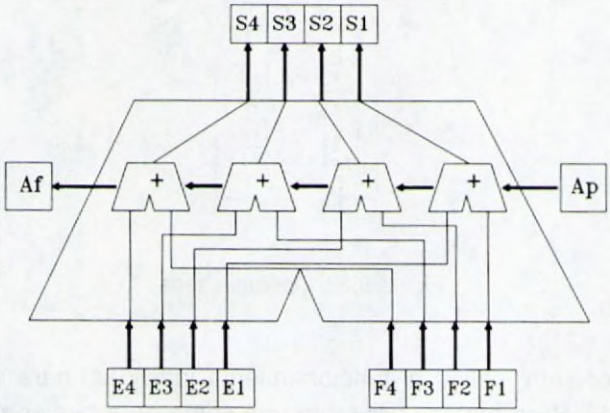


Figura 8.33. Operador paralelo

Este operador aritmético-lógico puede realizar también operaciones de multiplicación y división si se conectan registros auxiliares como los indicados en las figuras 8.34 y 8.35. La multiplicación se realiza mediante sucesivas etapas de suma-desplazamiento mientras que la división ejecuta etapas de resta-desplazamiento.

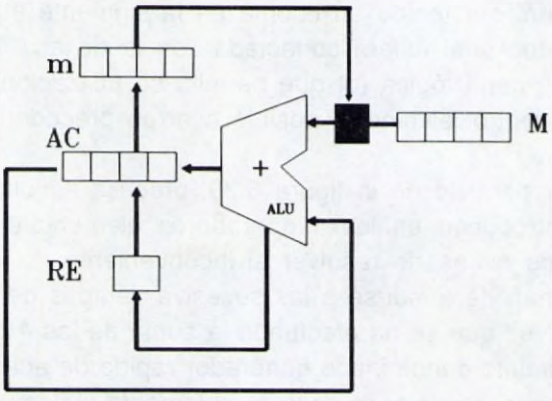


Figura 8.34. Operador de multiplicación

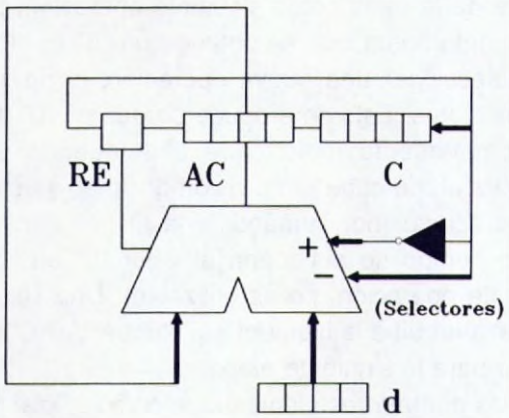


Figura 8.35. Operador de división

El registro "m" se utiliza por el multiplicador, "M" por el multiplicando y AC y RE son respectivamente el acumulador y el registro de estado. Entre "m" y "M" se establece una comparación de datos a través de un bloque de interconexión que puede ser un elemento triestado o una sencilla puerta lógica AND. Si el bit menos significativo de "m" es un "1", se transmite el valor de "M" para ser sumado con el acumulador. Si el bit menos significativo de "m" es un "0", el bloque de interconexión se bloqueará sumándose el acumulador con "0". Después de cada suma se desplazarán los bits de RE-AC-m perdiéndose el bit menos significativo de "m" y poniéndose a cero RE. El resultado final queda almacenado en la unión de los registros AC y "m".

Para realizar la división, las restas se realizarán mediante sumas de los complementos del sustraendo (por ejemplo complemento a 2 para no tener que realimentar el acarreo final), siendo por tanto necesario que el circuito conozca si el minuendo es mayor o no que el sustraendo. El registro "d" almacena el divisor y los registros "AC" y "C" almacenan el dividendo, alineando los bits hacia la derecha tanto en el conjunto "AC-C" como en "d" (se alinean a la izquierda si existe un contador de desplazamientos), quedando el resultado final en el registro "C" (Cociente) y en "AC" (Resto). Para comprender mejor el funcionamiento de este circuito conviene NO UTILIZAR EL BIT MENOS SIGNIFICATIVO DEL REGISTRO C.

Al comenzar la operación el circuito ejecuta una operación de resta entre "AC" y "d". Si se obtiene un "1" en el acarreo final (RE), se conoce que el minuendo es mayor que el sustraendo almacenándose el valor de "RE" en el bit menos significativo de "C" (un "1" en el cociente), se desplazan todos los bits de "AC-C" una posición hacia la izquierda, perdiéndose el bit más signifi-

cativo y se selecciona la resta como siguiente operación. Esta operación se repetirá sucesivamente hasta que se obtenga un "0" en "RE". En este caso, se procederá a seleccionar una nueva operación, la de suma a través del inversor de los selectores. Esto se produce porque un "0" de acarreo final en una suma en complemento indica que el minuendo es menor que el sustraendo (equivale al "no cabe" en división decimal) y es necesario corregir el nuevo valor del acumulador sumándole la misma cantidad que se le ha restado, al mismo tiempo se almacena el valor "0" en "C". (Si se obtiene acarreo final en esta operación, no es utilizable). Una vez corregido el acumulador se desplaza un bit a la izquierda el bloque "AC-C" y se selecciona la operación de resta para la siguiente etapa.

Si se unen varios operadores elementales para realizar distintas operaciones aritméticas y lógicas seleccionadas por señales de control, estamos ante el circuito denominado operador combinacional, auténtica unidad aritmética-lógica programable. El circuito de la figura 8.36 permite programar hasta 256 operaciones según las combinaciones binarias de sus entradas (en este caso no todas serían diferentes).

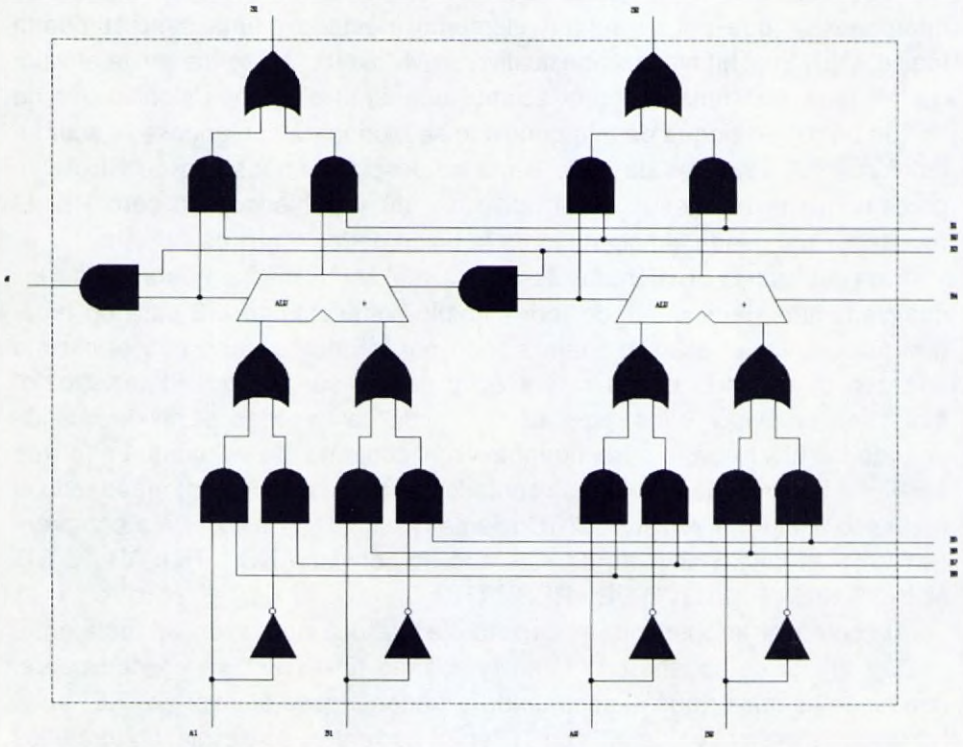


Figura 8.36. Operador combinacional de 2 bits

Algunos ejemplos de operaciones programables en el operador de la figura 8.36 pueden ser las siguientes:

E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	OPERACION
1	0	0	0	0	0	0	1	A
1	0	1	0	0	1	0	1	A+B
1	0	1	1	1	0	0	1	A-B
1	0	1	1	0	0	0	1	A+1
1	0	1	0	1	1	0	1	A-1
1	1	0	0	0	1	0	1	AUB
1	0	1	1	0	1	1	0	B-A

También los operadores pueden construirse atendiendo al sistema de numeración que se vaya a utilizar o la representación de los dígitos binarios. Por ejemplo en la figura 8.37 se construye una célula básica para un sumador en BCD donde se incluye además de la suma en BCD de los números originales, la corrección para aquellas cifras que excedan en dicha suma del valor máximo representativo del sistema de numeración.

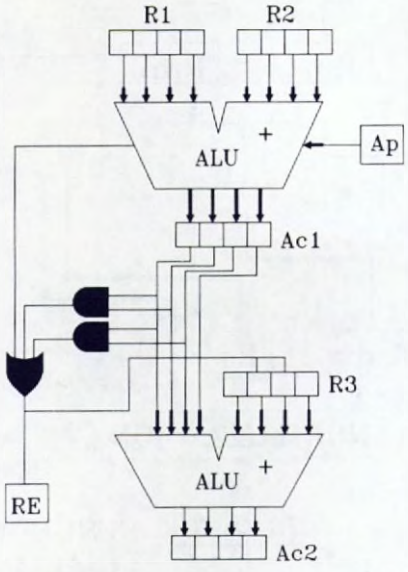


Figura 8.37. Operador BCD

Este circuito es asemejable a los correspondientes a otras variantes del sistema BCD, tal como los códigos Aiken y Exceso-3, y otros de características similares. El resultado obtenido necesitará una posterior transformación en el caso de números negativos y en el tratamiento de los acarreos finales, situación que se reauelve de idéntica forma que en los circuitos sumadores.

Como operador representativo de la codificación en binario y de su representación particular, se puede hacer mención del oparador en coma flotante, circuito de un nivel muy elevado, que opera con números transformados a coma flotante y con mantisa normalizada, siendo su coste muy alto en comparación con los operadores elementales. En la figura 8.38 se representa un circuito genérico que puede servir de aplicación a cualquier formato.

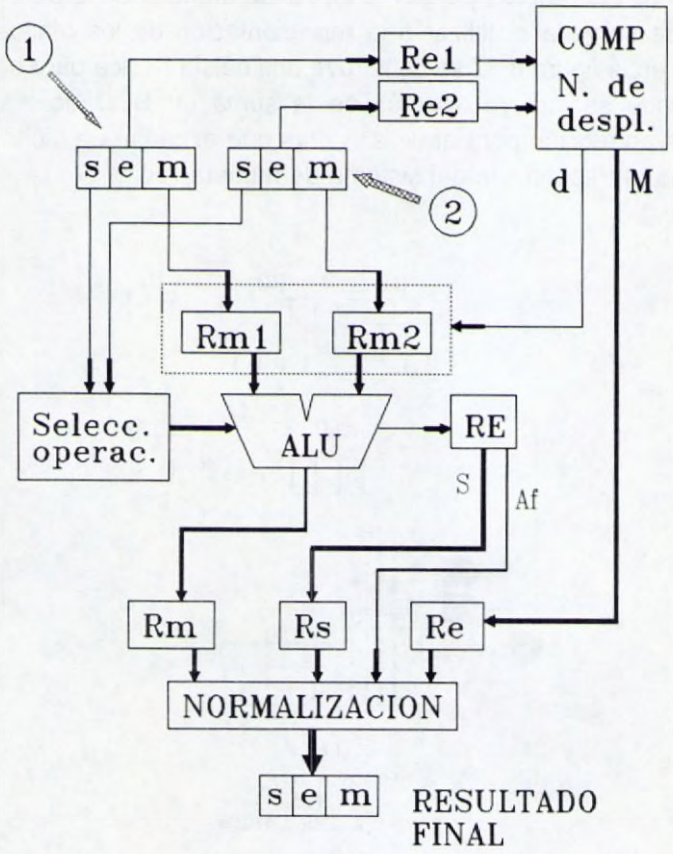


Figura 8.38. Operador en coma flotante

Los datos de entrada se almacenan en los registros "1" y "2", trasvasándose los exponentes a Re_1 y Re_2 y las mantisas a Rm_1 y Rm_2. La diferencia entre los exponentes determina el mayor de ellos y el número de desplazamientos que se efectuará sobre la mantisa del exponente inferior para equiparar mantisas con exponentes iguales.

Los signos de ambas cifras determinan la operación a efectuar en el operador, obteniéndose un resultado final que se almacena en los registros Rm (mantisa), Rs (signo) y Re (exponente), introduciéndose también la señal acarreo final procedente del registro de estado RE si llegara a existir. Con estos datos se procede a una normalización, obteniéndose un resultado final de características similares a los datos de entrada. Los registros auxiliares permiten efectuar simultáneamente varias operaciones ya que al pasar la información de unos registros a otros, un nuevo valor puede sustituirla, proporcionando así una gran velocidad de cálculo.

*## **Ejemplo 8.63**: Realizar la multiplicación de los números decimales 12 y 13 utilizando el operador de suma-desplazamiento.*

Solución:

RE	AC	m	M	Operación en AC	Acción
0	0000	0000	0000		P.Cero
0	0000	1101	1100		Carga
0	1100	1101	1100	0000+1100	Suma
0	0110	0110	1100		Desplazamiento
0	0110	0110	1100	0110+0000	Suma
0	0011	0011	1100		Desplazamiento
0	1111	0011	1100	0011+1100	Suma
0	0111	1001	1100		Desplazamiento
1	0011	1001	1100	0111+1100	Suma
0	1001	1100	1100		Desplazamiento

Resultado final: $10011100_2=156_{10}$

*## **Ejemplo 8.64**: Realizar la multiplicación de los números decimales 37 y 19 utilizando el operador de suma-desplazamiento con registros de 8 bits.*

Solución:

RE	AC	m	M	Acción
0	00000000	00000000	00000000	P.Cero
0	00000000	00100101	00010011	Carga
0	00010011	00100101	00010011	Suma
0	00001001	10010010	00010011	Desplazamiento
0	00001001	10010010	00010011	Suma
0	00000100	11001001	00010011	Desplazamiento
0	00010111	11001001	00010011	Suma
0	00001011	11100100	00010011	Desplazamiento
0	00001011	11100100	00010011	Suma
0	00000101	11110010	00010011	Desplazamiento
0	00000101	11110010	00010011	Suma
0	00000010	11111001	00010011	Desplazamiento
0	00010101	11111001	00010011	Suma
0	00001010	11111100	00010011	Desplazamiento
0	00001010	11111100	00010011	Suma
0	00000101	01111110	00010011	Desplazamiento
0	00000101	01111110	00010011	Suma
0	00000010	10111111	00010011	Desplazamiento

Resultado final: $0000001010111111_2=703_{10}$

*## **Ejemplo 8.65**: Realizar la multiplicación de los números decimales 5 y 6 utilizando el operador de suma-desplazamiento con registros de 4 bits.*

Solución:

RE	AC	m	M	Operación en AC	Acción
0	0000	0000	0000		P.Cero
0	0000	0101	0110		Carga
0	0110	0101	0110	0000+0110	Suma
0	0011	0010	0110		Desplazamiento
0	0011	0010	0110	0011+0000	Suma
0	0001	1001	0110		Desplazamiento
0	0111	1001	0110	0001+0110	Suma
0	0011	1100	0110		Desplazamiento
0	0011	1100	0110	0011+0000	Suma
0	0001	1110	0110		Desplazamiento

Resultado final: $00011110_2=30_{10}$

*## **Ejemplo 8.66**: Efectuar la división de los números decimales 89 y 9 utilizando el operador de resta-desplazamiento. (Para mayor claridad se indicará con un guión la posición donde se almacenará el siguiente valor de "C", aunque este guión siempre equivale a "0").*

Solución:

RE	AC	C	d	Operación en AC	Acción
0	0000	0000	0000		P.Cero
0	1011	001 -	1001		Carga
1	0010	001 1	1001	1011+0111	Resta/Carga de C
0	0100	01 1-	1001		Desplazamiento
0	1011	01 10	1001	0100+0111	Resta/Carga de C
1	0100	01 10	1001	0011+1001	Corrección
0	1000	1 10-	1001		Desplazamiento
0	1111	1 100	1001	1000+0111	Resta/Carga de C
1	1000	1 100	1001	1111+1001	Corrección
1	0001	100-	1001		Desplazamiento
1	1000	1001	1001	10001+0111	Resta

Resultado final: $1000\ 1001_2$ Resto=8 Cociente=9

*## **Ejemplo 8.67**: Efectuar la división de los números decimales 60 y 10 utilizando el operador de resta-desplazamiento. (Para mayor claridad se indicará con un guión la posición donde se almacenará el siguiente valor de "C", aunque este guión siempre equivale a "0").*

Solución:

RE	AC	C	d	Operación en AC	Acción
0	0000	0000	0000		P.Cero
0	0111	100 -	1010		Carga
0	1101	100 0	1010	0111+0110	Resta/Carga de C
1	0111	100 0	1010	1101+1010	Corrección
0	1111	00 0-	1010		Desplazamiento
1	0101	00 01	1010	1111+0110	Resta/Carga de C
0	1010	0 01-	1010		Desplazamiento
1	0000	0 011	1010	1010+0110	Resta/Carga de C
0	0000	011-	1010		Desplazamiento
0	0110	0110	1010	0000+0110	Resta/Carga de C
1	0000	0110	1010	0110+1010	Corrección

Resultado final: $0000\ 0110_2$ Resto=0 Cociente=6

*## **Ejemplo 8.68**: Sumar los números decimales 8 y 9 con el operador BCD.*

Solución:

RE	AC2	R3	AC1	R1	R2	Ap	
0	0000	0000	0000	0000	0000	0	P.Cero
0	0000	0000	0000	1000	1001	0	Carga de R1 y R2
1	0000	0000	0001	1000	1001	0	Carga de AC1
1	0000	0110	0001	1000	1001	0	Carga de R3
1	0111	0110	0001	1000	1001	0	Carga de AC2

Resultado final: RE=1 CAC2=0111 17_{10}

*## **Ejemplo 8.69**: Sumar en coma flotante los números 0/01100/1011000000 y 0/01000/1110000000 en formato s/e/m.*

Solución:

Diferencia de exponentes: d=4

Exponente mayor: 01100

Mantisas después del desplazamiento:

Rm1=1011000000

Rm2=0000111000

Suma de mantisas: 1011111000

RE=0 (Resultado ya normalizado)

Rs=0

Resultado final: 0 01100 1011111000

*## **Ejemplo 8.70**: Sumar en coma flotante los números 0/01100/1111110100 y 0/01000/1110000000 en formato s/e/m.*

Solución:

Diferencia de exponentes: d=4

Exponente mayor: 01100

Mantisas después del desplazamiento:

Rm1=1111110100

Rm2=0000111000

Suma de mantisas: 0000101100

RE=1

Rs=0

Resultado final: 0 01101 1000010110

8.16. APENDICE - CONCEPTOS BASICOS DE DISEÑO DE CIRCUITOS DE CONTROL

Al aumentar la complejidad de los sistemas digitales, tienen que ser analizados desde un punto de vista diferente al puramente combinacional ya que intervienen numerosos factores en su diseño, apareciendo entonces la necesidad de implantar un elemento que controle los diversos elementos, circuitos, operaciones, etc., asignando prioridades, estableciendo activaciones y desactivaciones, poniendo a cero y permitiendo el traspaso de información. Este elemento va a ser definido como "Unidad de Control" y suele ser particular para cada circuito o sistema digital que se vaya a construir y diseñar.

Se analizan a continuación los circuitos de control que se pueden construir a base de elementos simples de la lógica combinacional tales como puertas lógicas, biestables, registros, contadores, etc., lo que se denominará lógica cableada, diferenciándola de la lógica programada constituida por bloques fijos que pueden ser adaptados a diversas aplicaciones. Estos pasos y esquemas sencillos que se describen, aplicados a circuitos sumadores, operadores, etc., servirán de introducción al análisis de unidades de control más complejas que constituyen el centro de los sistemas basados en microprocesadores (Formados por un bloque de proceso basado en operadores combinacionales y una unidad de control con registros auxiliares de memoria y trabajo). Para mejor comprensión de este apartado se requiere conocer los temas de biestables, contadores y registros.

Para proceder al ensamblaje de los elementos del circuito de control es necesario realizar el desarrollo previo de las ecuaciones que definen dichos elementos, de forma equivalente a la constitución de las ecuaciones lógicas que definen los sistemas combinacionales. Sin embargo estas ecuaciones serán distintas y necesitan una descripción previa de algunos conceptos que intervienen en su contexto:

- Los circuitos que se diseñarán serán síncronos, adaptando por tanto los elementos que se utilicen a dicha característica.
- El circuito funcionará a partir de una señal externa de puesta en marcha.
- Existirá siempre un biestable que indicará al resto del sistema que está activado el circuito de control.
- Del circuito de control saldrán las señales necesarias de carga, puesta a cero, desplazamiento, etc. que serán de aplicación a los elementos del circuito operativo.
- Opcionalmente se pueden añadir entradas externas de Inhibición (Paraliza momentáneamente la operación), Continuación (Reanuda una operación detenida momentáneamente) y Puesta a cero (Inicializa la operación en

cualquier instante) que puede necesitar o no una posterior acción de puesta en marcha según se desee diseñar.

Las ecuaciones de diseño tendrán una estructura compuesta por una igualdad que equivale a una secuencia de actuaciones o conjunto acción-reacción más que a una equivalencia de términos en sentido matemático. La primera parte de la igualdad estará constituida por los siguientes aspectos:

- ACCIONES EXTERNAS aplicadas al circuito de control, suponiendo únicamente como básica la puesta en marcha

- ACCIONES INTERNAS generadas por el propio circuito de control, como la señal interna de activación o señales de decodificación, cuenta o sincronía

- ENTRADAS PROCEDENTES DEL CIRCUITO DE PROCESO al cuál se aplica el circuito de control

La segunda parte de dicha expresión estará integrada por las siguientes reacciones sobre los elementos del sistema:

- ACTUACIONES DEL CIRCUITO DE CONTROL sobre el proceso que se estudia
- ACTUACIONES DEL CIRCUITO DE CONTROL sobre los componentes del propio circuito de control

(A. Externas)(A. Internas)(E. Proceso) => Act. Proceso, Act. Internas

Los elementos de la primera parte de la ecuación representan terminales de los circuitos, mientras que las actuaciones de la segunda parte representan la evolución que tendrán las entradas o salidas de los elementos. Las iniciales, siglas y expresiones especiales que se emplearán en los ejercicios que se desarrollan a continuación son las siguientes:

PM=Puesta en marcha del circuito de control
BA=Señal de activación del circuito de control
PC=Puesta a cero de un registro
C=Carga de un registro
B1, B2=Biestables auxiliares
C1, C2, C3 ...=Señales de control
RE=Registro de estado

R1, R2, R3 ...=Registros
AC=Registro acumulador
IN=Señal de inhibición o interrupción momentánea
CN=Señal de continuación después de activar IN
PCE=Puesta a cero exterior en cualquier instante

Los ejemplos analizan y describen un procedimiento de diseño que conlleva a la construcción de un circuito que controle el proceso en estudio, describiendo conceptos genéricos y resaltando los aspectos que intervienen en el mismo. Sin embargo, en relación con los esquemas reales y para evitar que aparezcan dibujos difíciles de comprender y analizar, sólo se han representado las señales, conexiones y terminales más importantes para realizar las operaciones elementales. Las ecuaciones básicas tradicionales en la lógica combinacional serán modificadas introduciendo señales de control que permitirán que las mismas se realicen o no. Estas señales se aplicarán normalmente a las operaciones de cargar datos, complementar, desplazar, poner a cero, etc.

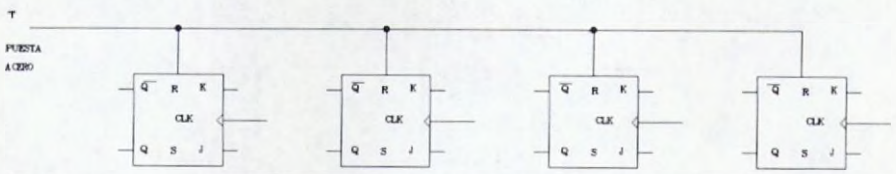


Figura 8.39. Puesta a cero

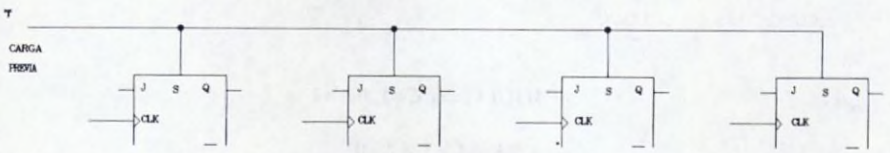


Figura 8.40. Carga previa

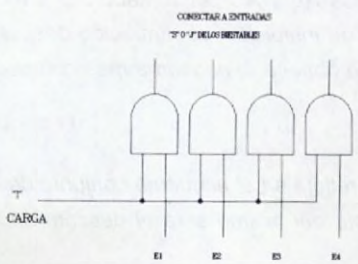


Figura 8.41. Carga

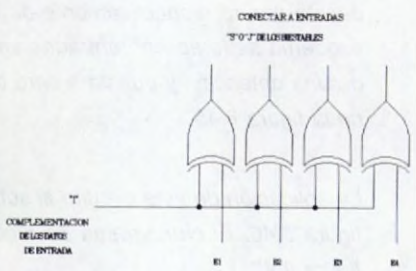


Figura 8.42. Complementación

*## **Ejemplo 8.71**: Diseñar el circuito de control de un operador de suma compuesto por dos registros de entrada de datos A y B y un registro de salida AC con registro de estado RE para el acarreo.*

Solución:

Se supone un esquema simple como el representado en la figura 8.43 donde los registros de entrada de datos A y B se suponen cargados desde el exterior antes de comenzar el proceso de control, para simplificar así el diseño del mismo.

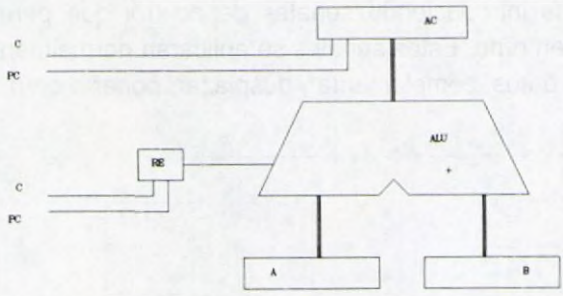


Figura 8.43

Ecuaciones de control:

$$(PM)(\overline{BA}) \Rightarrow PC=1, BA=1$$

$$BA \Rightarrow C=1, BA=0$$

La expresión PC=1 indica que se realiza AC=0 y RE=0 mientras que C=1 indica que la ALU realiza la operación AC=A+B. Las salidas C y PC son la carga y puesta a cero respectivamente de los registros RE y AC del sumador. Si a este esquema se le aplican entradas adicionales de inhibición, continuación después de una detención y puesta a cero externa, se obtiene el esquema más completo de la figura 8.45.

La aplicación de este circuito al sumador se refleja en el esquema conjunto de la figura 8.46. El cronograma de funcionamiento del mismo será el descrito en la figura 8.47.

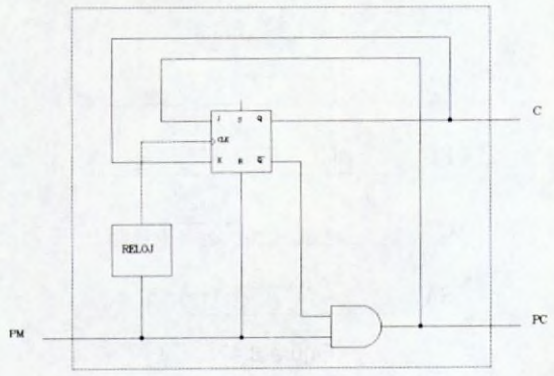


Figura 8.44

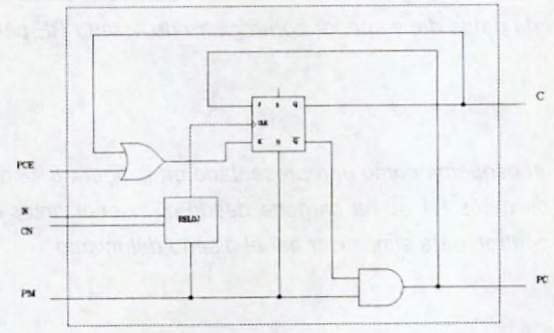


Figura 8.45

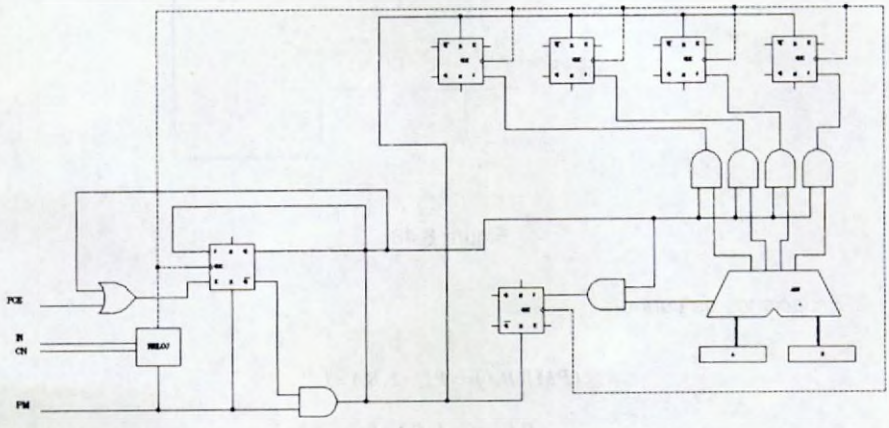


Figura 8.46

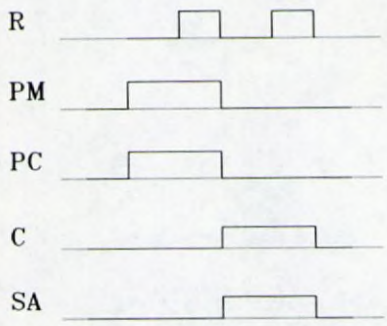


Figura 8.47

*## **Ejemplo 8.72**: Diseñar un circuito de control de un operador de suma compuesto por un registro R1 de entrada de datos y un registro de salida acumulador AC que se realimenta a la segunda entrada de datos del sumador con registro de estado RE para el acarreo.*

Solución:

Se supone un esquema como el representado en la figura 8.48 donde el registro de entrada de datos R1 se ha cargado desde el exterior antes de comenzar el proceso de control, para simplificar así el diseño del mismo.

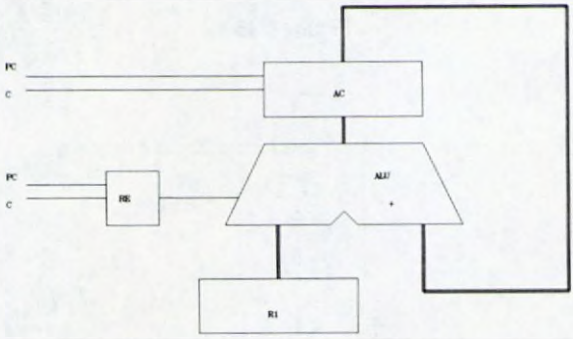


Figura 8.48

Ecuaciones de control:

$$(PM)(\overline{BA}) \Rightarrow PC=1, BA=1$$

$$BA \Rightarrow C=1, BA=0$$

Estas ecuaciones son iguales que las del ejemplo anterior por lo que el circuito de control será el mismo que el realizado para dicho ejemplo. Analizando este caso, se saca como conclusión que este circuito realiza la operación de sumar el valor del registro R1 con cero y almacenarlo en el acumulador. Esta podría ser la primera parte de la operación de suma de dos términos, necesitando ahora el registro R1 una segunda entrada de datos para que sean sumados con los que ya existen en el acumulador.

***## Ejemplo 8.73**: Diseñar el circuito de control de un operador de suma compuesto por un registro R1 de entrada de datos y un registro de salida acumulador AC que se realimenta a la segunda entrada de datos del sumador con registro de estado RE para el acarreo, disponiendo de una entrada de control de la entrada de datos al registro R1.*

Solución:

Se supone un esquema como el representado en la figura 8.49 donde el registro de entrada de datos R1 se carga a través de una entrada externa de datos controlada por la señal de control C1.

Ecuaciones de control:

$$(PM)(\overline{BA}) \Rightarrow PC=1, BA=1, B1=0$$

$$(BA)(\overline{B1}) \Rightarrow C1=1, B1=1$$

$$(BA)(B1) \Rightarrow C=1, BA=0$$

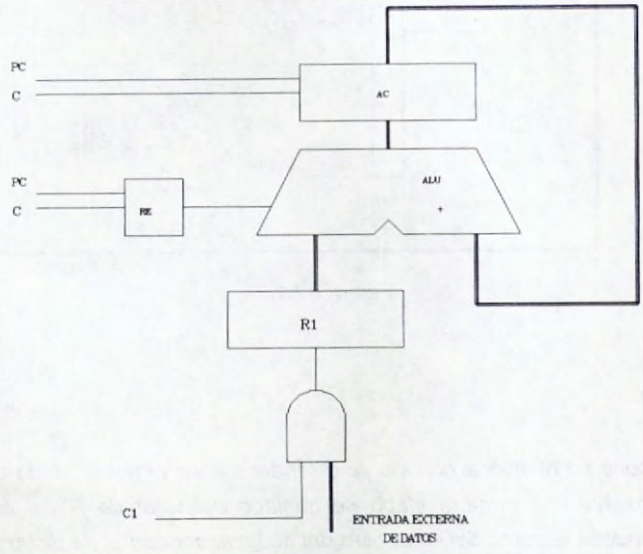


Figura 8.49

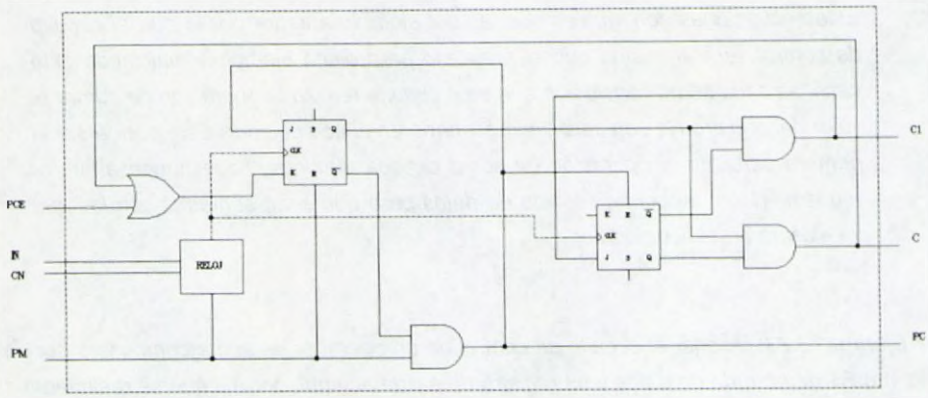


Figura 8.50

*## **Ejemplo 8.74**: Diseñar el circuito de control del sumador de la figura 8.51, que toma los elementos a sumar de los registros R1 y R2 y almacena el resultado final en el registro R3 (además del RE). Se supone que los datos a utilizar en el proceso de suma ya están almacenados en los registros correspondientes.*

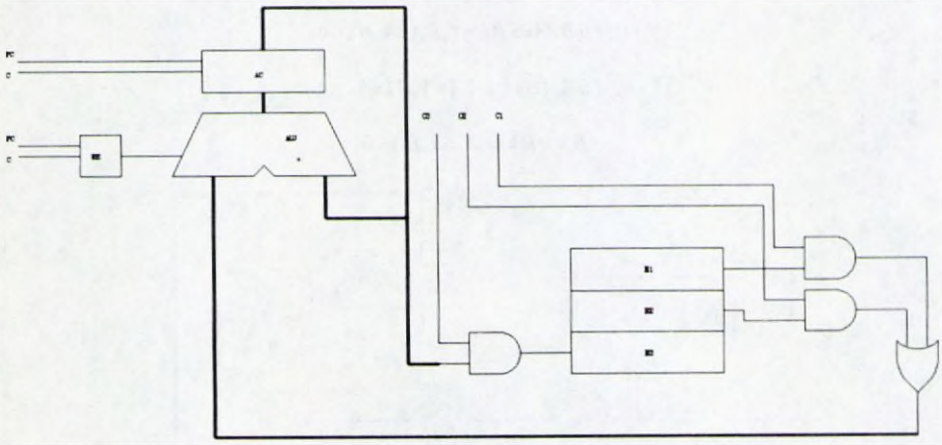


Figura 8.51

Solución:

Este circuito se relaciona con los ya diseñados anteriormente, dado que tendrá que realizarse una primera etapa de traslado del valor de R1 al acumulador produciéndose la suma del valor cero del acumulador con el de dicho registro. A continuación se procederá a sumar el nuevo valor del acumulador con el del

registro R2, almacenándose el resultado final en AC. La última etapa será la de proceder a trasladar el resultado de AC a un registro auxiliar R3.

Este esquema puede ser un modelo sencillo de una etapa de trabajo de un microprocesador, compuesta por una unidad aritmético-lógica, un circuito de control y un bloque de registros auxiliares y memoria. El análisis se realiza teniendo en cuenta que se establecen tres etapas o fases de trabajo además de la inicial de puesta a cero, por lo que se utilizarán dos biestables para definir dichas situaciones.

Se pueden utilizar contadores para simular estas situaciones o estados de trabajo del circuito, aunque en este caso es algo más simple el circuito con biestables ya que necesita sólo 3 estados diferentes.

Ecuaciones de control:

$$(PM)(\overline{BA}) \Rightarrow PC=1, BA=1, B1=0, B2=0$$

$$(BA)(\overline{B1})(\overline{B2}) \Rightarrow C=1, C1=1, B1=1, B2=0$$

$$(BA)(B1)(\overline{B2}) \Rightarrow C=1, C2=1, B2=1$$

$$(BA)(B1)(B2) \Rightarrow C3=1, BA=0$$

Se han utilizado las combinaciones "00", "10" y "11" de los biestables B1 y B2 aleatoriamente, pudiéndose emplear cualquier otra combinación de estados que conlleve a la activación de las tres señales necesarias. La realización física de estas ecuaciones conduce al circuito mostrado en la figura 8.52.

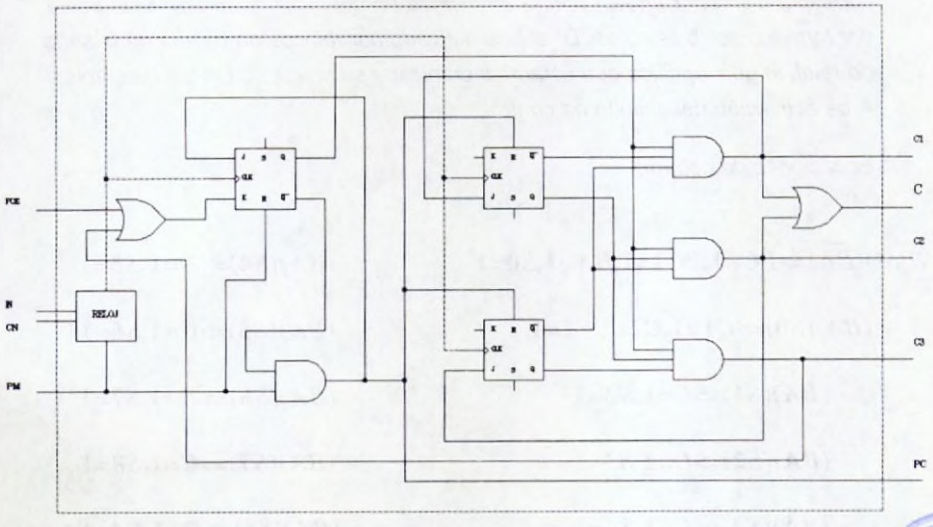


Figura 8.52

*## **Ejemplo 8.75**: Diseñar el circuito de control de un multiplicador de sumas-desplazamientos según el esquema representado en la figura 8.53.*

Solución:

Este sistema dispone de dos entradas externas de datos que rellenarán los registros correspondientes al multiplicando y al multiplicador, controlados por ambas señales de control. El conjunto formado por el registro de estado, el acumulador y el multiplicador servirá para almacenar los resultados parciales y el resultado final, aplicándoseles una señal de desplazamiento de la información que contienen.

Al mismo tiempo, el bit menos significativo del registro multiplicador permitirá que el multiplicando sea o no sumado al valor del acumulador en cada una de las etapas de suma. Podría conectarse en dicha puerta AND de entrada al sumador otra señal de control, pero no se ha considerado imprescindible para su correcto funcionamiento.

Las señales de puesta a cero se han desglosado en dos, una que pone a cero todos los registros excepto el registro de estado y otra específicamente para dicho registro, de aplicación particular a RE por si se quiere aplicar separadamente. Este ejemplo se va a diseñar con un contador ya que hay mayor número de etapas de operación (utilizando registros de 4 bits por ej.).

El contador a emplear puede ser un contador conmutado en cola de Johnson que en cada etapa de cuenta activa una salida diferente a través del decodificador correspondiente. (También podría emplearse un contador de anillo). Este contador formado por biestables "D" estará activado por flancos de bajada de la señal de reloj, lo que significa que estará sincronizada su salida con la del biestable J-K de activación del circuito de control.

Ecuaciones de control:

$$(PM)(\overline{BA}) \Rightarrow PC=1, PC1=1, BA=1, S0=1$$

$$(BA)(S0) \Rightarrow C1=1, C2=1, S1=1$$

$$(BA)(S1) \Rightarrow C=1, S2=1$$

$$(BA)(S2) \Rightarrow D=1, S3=1$$

$$(BA)(S3) \Rightarrow C=1, S4=1$$

$$(BA)(S4) \Rightarrow D=1, S5=1$$

$$(BA)(S5) \Rightarrow C=1, S6=1$$

$$(BA)(S6) \Rightarrow D=1, S7=1$$

$$(BA)(S7) \Rightarrow C=1, S8=1$$

$$(BA)(S8) \Rightarrow D=1, BA=0$$

Aunque en las ecuaciones se ha indicado el incremento del contador, no es imprescindible hacerlo ya que el contador se incrementa automáticamente. En la figura 8.54 se representa la estructura de este circuito de control, cuyo funcionamiento se analiza en el cronograma de la figura 8.55. Este circuito puede mejorarse sincronizando las señales de desplazamiento con la señal de reloj, ejecutándose en el mismo ciclo que la señal de carga ya que con la mitad del tiempo empleado por una señal de reloj se puede modificar el valor de un biestable. De esta forma el proceso completo tardaría 4 señales menos de reloj.

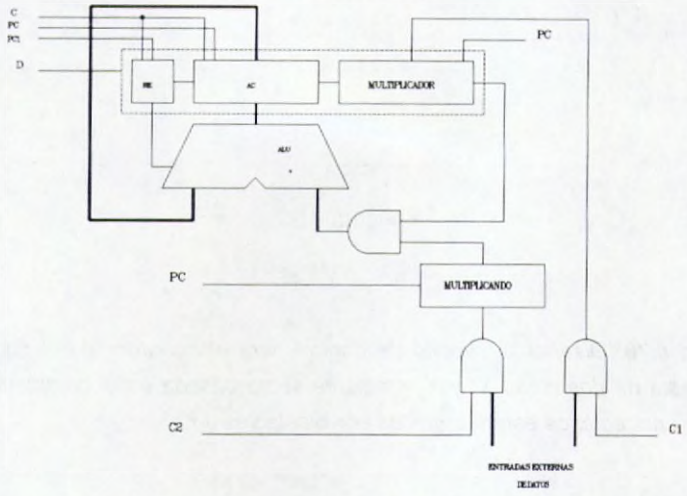


Figura 8.53

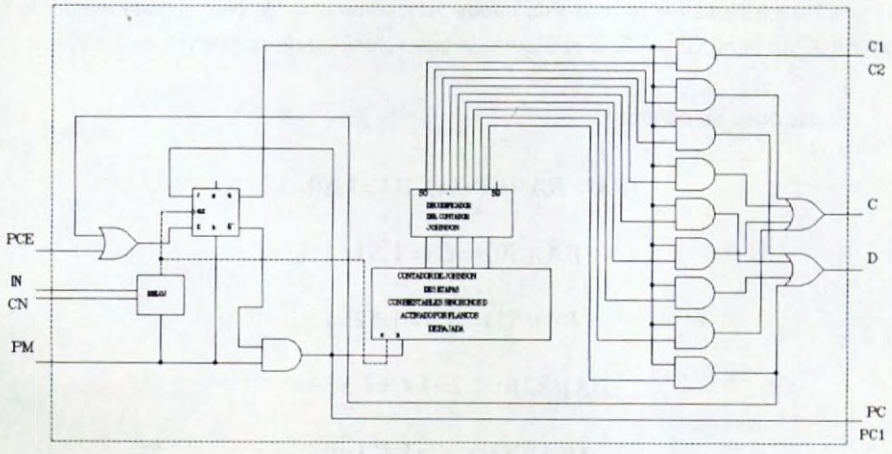


Figura 8.54

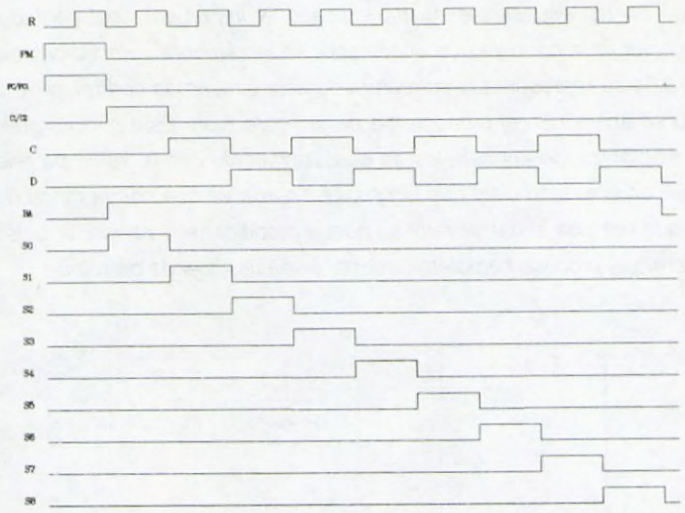


Figura 8.55

*## **Ejemplo 8.76**: Diseñar el circuito de control para el operador de la figura 8.56 que realiza la resta de dos cifras X_1 y X_2 mediante suma basada en el complemento a 1 del sustraendo. Los registros están formados por biestables J-K.*

Solución:

Se supone que los datos X_1 y X_2 son proporcionados por otro circuito. El proceso comienza una vez puestos a cero los registros, con el dato X_1 en el registro R3. Cuando dicha información se traspasa al registro R2, puede suponerse que también simultáneamente se introduce X_2 en el registro R3, lo que permite una velocidad de operación muy elevada.

Ecuaciones de control:

$$(PM)(\overline{BA}) \Rightarrow PC=1, BA=1, S0=1$$

$$(BA)(S0) \Rightarrow C1=1, S1=1$$

$$(BA)(S1) \Rightarrow C1=1, S2=1$$

$$(BA)(S2) \Rightarrow C2=1, C=1, S3=1$$

$$(BA)(S3) \Rightarrow C=1, BA=0$$

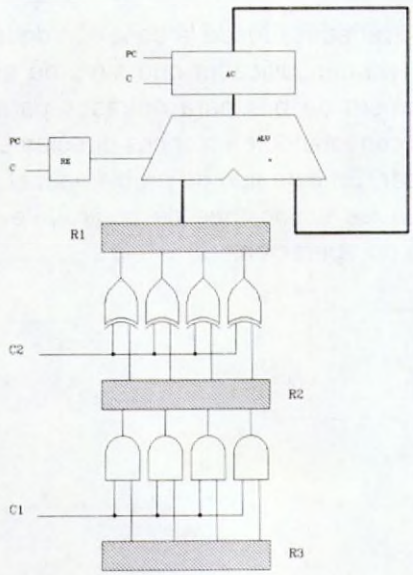


Figura 8.56

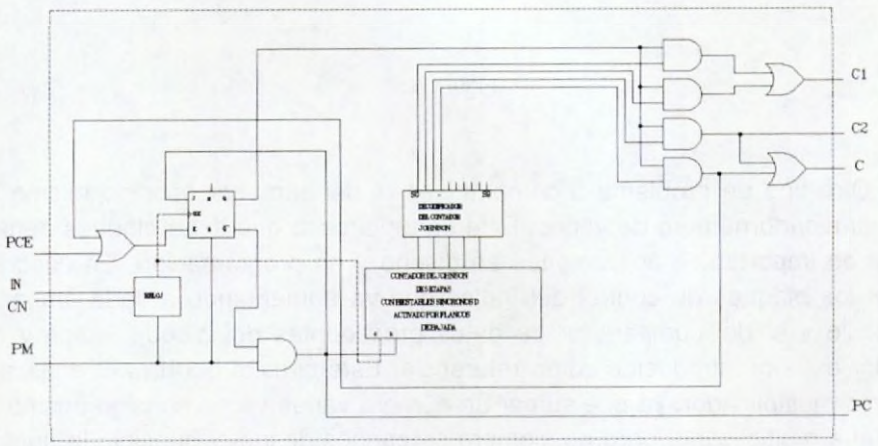


Figura 8.57

Es interesante comentar el diseño de un circuito de control para el esquema del multiplicador teniendo en cuenta que el número de bits del registro multiplicador puede variar no teniendo por qué ser igual a cuatro. Se puede indicar que en este caso, la línea de entrada de datos hacia el multiplicador, tendrá también una conexión hacia el interior del circuito de control para conocer en cada opción el número de bits que se introducen en dicho regis-

tro. Las soluciones son variadas, desde la conexión de las salidas del registro multiplicador (4 bits) a un decodificador que sirve de entrada a un contador que almacenará el número de bits para entradas paralelas, o utilizando la entrada serie de datos conectándolo entonces directamente a un contador sin necesidad de decodificar. En este tipo de problemas, el primer término de las igualdades que definen las ecuaciones de control, llevará incluida la señal procedente del circuito de operación.

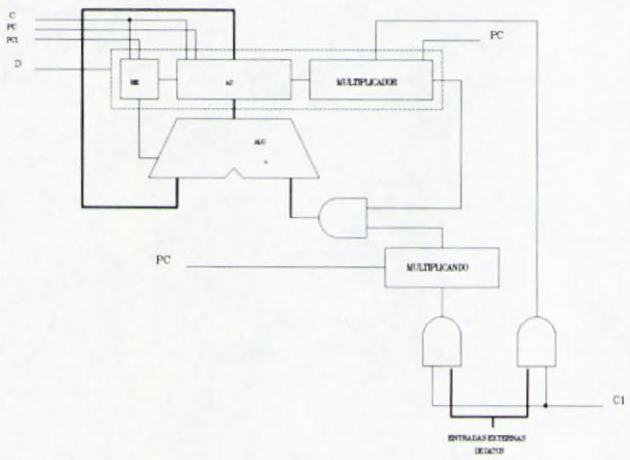


Figura 8.58

Otro tipo de problema a comentar es el del sumador consigo mismo un determinado número de veces. Este es un circuito que tiene bloques genéricos de importantes aplicaciones en diseño y en programación. En concreto son los bloques de control del índice que va aumentando a cada etapa de cálculo y el del comparador de datos procedentes del bloque índice y del valor exterior introducido como referencia. Este circuito podría ser analizado como multiplicador, ya que sumar un número varias veces consigo mismo es igual a multiplicarlo, pero no siempre se conoce desde el principio la necesidad de tener que multiplicar un determinado dato.

También sumar una cantidad idéntico número de veces consigo mismo equivale a elevar al cuadrado. Sin embargo es más sencillo utilizar el circuito multiplicador para elevar al cuadrado, conectando los registros multiplicador y multiplicando a las mismas entradas de datos y de control como se indica en la figura 8.58. (Otros problemas interesantes pueden ser el circuito de control de un divisor, el de un operador que sume un número "n" veces consigo mismo, el de un operador combinacional, el de un operador BCD y el de un operador en coma flotante).

En conjunto y como resumen final, se puede añadir que el diseño de un circuito es variado, dependiendo de los componentes que se conozcan y de los que se dispongan, existiendo alternativas que producen idénticos resultados. El análisis realizado de describir etapas de funcionamiento de un circuito llevará al diseño de un circuito generador de etapas o secuencias que controlará de forma más cómoda los circuitos de control y que permitirá diseños más simples de circuitos mucho más complicados como son los microprocesadores.

TEMA 9
BIESTABLES

9

BIESTABLES

9.1. INTRODUCCION

En el presente tema se analizará un tipo particular de circuito lógico denominado biestable. Es un tema que se suele englobar en el estudio de los circuitos secuenciales, parte fundamental en el diseño de los autómatas. Sin embargo, dicho circuito no es otra cosa que un conjunto de puertas lógicas en su diseño digital, por lo que se puede analizar independientemente como un circuito combinacional.

Sin embargo, dado que existen diferentes modelos de biestables según su funcionamiento, se introducirá en este tema el concepto de señal de reloj que sincronizará los circuitos. De esta forma, se denominarán circuitos asíncronos los que sólo cambian sus valores de salida cuando cambian las condiciones del circuito, es decir, cuando cambian las variables de entrada y circuitos síncronos cuando cambian de acuerdo con la señal de sincronía, una señal que determina el momento de realizar la transición de una situación a otra. Si los valores de las variables cambian antes de que aparezca la señal de sincronía, no se transmitirán dichos cambios al circuito.

Los biestables almacenan un bit de información, siendo por tanto la célula básica de almacenamiento de información de los circuitos digitales. El tema completo describe el funcionamiento y las características de los diferentes biestables asíncronos y síncronos utilizados en circuitería digital.

9.2. CLASIFICACION DE LOS BIESTABLES

Según se ha indicado en la introducción, los biestables se clasifican en asíncronos y síncronos según su modo de funcionamiento. Son biestables asíncronos los que cambian sus salidas cuando tiene lugar un cambio en sus

entradas, evolucionando como cualquier circuito combinacional. Son biestables síncronos los que necesitan además del cambio en sus entradas, una señal de sincronía o señal de reloj.

Los biestables asíncronos están activados por niveles como los circuitos combinacionales analizados en temas anteriores mientras que los síncronos se activan por flancos (Ver introducción a señales digitales en capítulo 2).

Son biestables asíncronos los denominados "S-R", "T" y "LATCH", mientras que el grupo de los síncronos lo forman los biestables "D" y "J-K".

9.3. **EL BIESTABLE S-R**

El biestable S-R o **SET-RESET** es el más importante de los biestables asíncronos, sirviendo de base para la definición del resto de los biestables. (Existe también una vertiente de biestable S-R con reloj, pero se engloba en los modelos de biestables J-K). Posee 2 entradas de señal, denominadas S (set o conexión) y R (reset o desconexión).

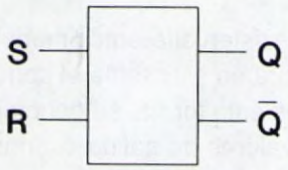


Figura 9.1

Su funcionamiento se define de forma que si S es "1" la salida es "1" y si R es "1" la salida es "0". Si las 2 entradas son simultáneamente "0", la salida no se modificará y si ambas entradas son "1", no existe definición de salida para el biestable. Esto quiere decir que no se puede dar la combinación de entradas "11". Si se simboliza la variable de salida como "Q", variable de evolución temporal, éste será el valor que existe en el terminal de salida del biestable un instante de tiempo antes de que se apliquen los valores correspondientes a las nuevas entradas, mientras que "Q(+)" representa el valor de dicha variable un instante de tiempo después de aplicar las mismas. Así por ejemplo, en la tabla 9.1 correspondiente a la definición del biestable S-R, si se analiza la cuarta línea, el valor de "Q" es "1". Quiere esto decir que antes de aplicar las entradas el valor de la salida es "1" y que una vez aplicadas las entradas S=0 y R=1 la salida tomará el valor Q(+)=0

un instante de tiempo después. Por tanto sólo existe un terminal "Q" analizado en dos instantes de tiempo diferentes.

S	R	Q	Q(+)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	-
1	1	1	-

Tabla 9.1. Biestable S-R

Aplicando simplificación por Karnaugh para funciones incompletamente especificadas a la tabla 9.1, se obtiene la siguiente ecuación:

$$Q=S+Q\bar{R} \qquad [9.1]$$

Los circuitos empleados como biestables S-R están formados a base de puertas NAND y NOR, por lo que mediante doble y cuádruple negación respectivamente se llega a la obtención de los circuitos de las figuras 9.2 y 9.3. Estos circuitos toman la salida complementada a través de la negación de la salida Q. Existen unas aplicaciones especiales de estos circuitos utilizando las combinaciones S=1 y R=1 en sus entradas. Estos circuitos se denominan de **INSCRIPCION PRIORITARIA** y de **BORRADO PRIORITARIO**.

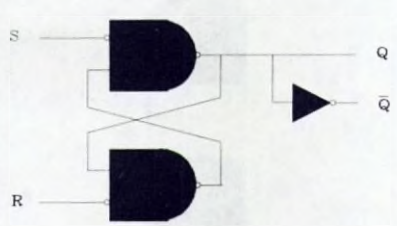


Figura 9.2. Biestable S-R con puertas NAND

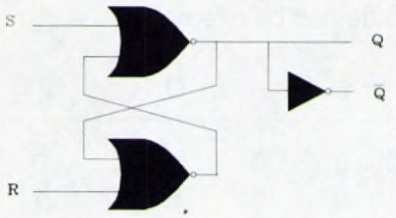


Figura 9.3. Biestable S-R con puertas NOR

El circuito denominado de INSCRIPCION PRIORITARIA (no es un biestable), utiliza parte de sus combinaciones de entrada para funcionar como biestable y además cuando ambas entradas sean iguales a "1" la salida siempre será "1". Su tabla de verdad representativa es la 9.2 y su circuito el de la figura 9.4. Este circuito rompe la lógica algebraica dado que para entradas iguales a "1" las salidas Q y su complemento son iguales.

S	R	Q	Q(+)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Tabla 9.2. Circuito de inscripción prioritaria

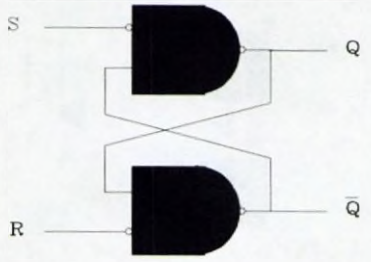


Figura 9.4. Circuito de inscripción prioritaria

Igualmente para el circuito de BORRADO PRIORITARIO se establecen las mismas premisas que para el circuito de inscripción, en este caso obteniendo un "0" a la salida siempre que las entradas sean iguales a "1" ya que se emplea el circuito de puertas NOR de la figura 9.5.

S	R	Q	Q(+)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Tabla 9.3. Circuito de borrado prioritario

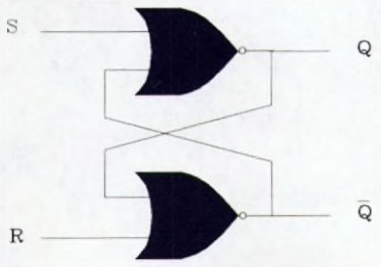


Figura 9.5. Circuito de borrado prioritario

La tabla general 9.1 del biestable S-R se puede agrupar y simplificar obteniendo la tabla general de representación de este biestable indicada en 9.4. Esta es la tabla que se emplea para constituir el resto de los biestables.

Q	Q(+)	S	R
0	0	0	-
0	1	1	0
1	0	0	1
1	1	-	0

Tabla 9.4. Biestable S-R

*## **Ejemplo 9.1**: Aplicar el cronograma de la figura 9.6 a las entradas de un biestable S-R y obtener gráficamente la salida.*

Solución:

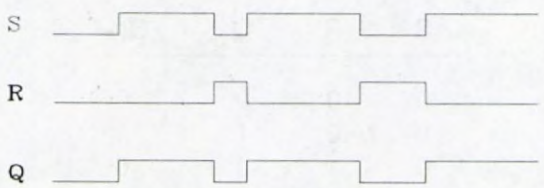


Figura 9.6

*## **Ejemplo 9.2**: Aplicar el cronograma de la figura 9.7 a las entradas de un circuito de inscripción prioritaria y obtener gráficamente la salida.*

Solución:

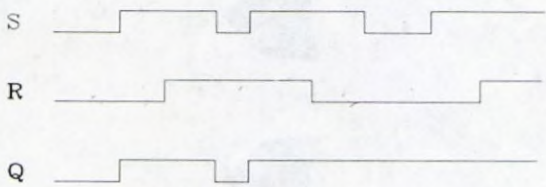


Figura 9.7

*## **Ejemplo 9.3**: Aplicar el cronograma de la figura 9.8 a las entradas de un circuito de borrado prioritario y obtener gráficamente la salida.*

Solución:

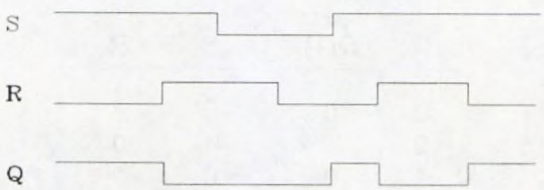


Figura 9.8

9.4. **EL BIESTABLE T**

Su denominación procede de la descripción inglesa "toggle". Es el biestable de conmutación, ya que cambia el valor de la salida cada vez que se aplica una señal (un pulso) a la entrada del mismo. Es decir, la salida cambia de "0" a "1" ó de "1" a "0" cuando la entrada T="1". Su esquema bloque se representa en la figura 9.9.

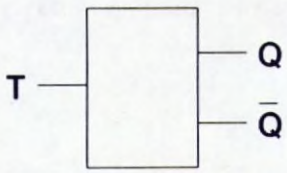


Figura 9.9. El biestable T

El biestable T funciona de acuerdo con la siguiente tabla, analizada en base a los valores de la salida:

T	Q	Q(+)
0	0	0
0	1	1
1	0	1
1	1	0

Tabla 9.5. Biestable T

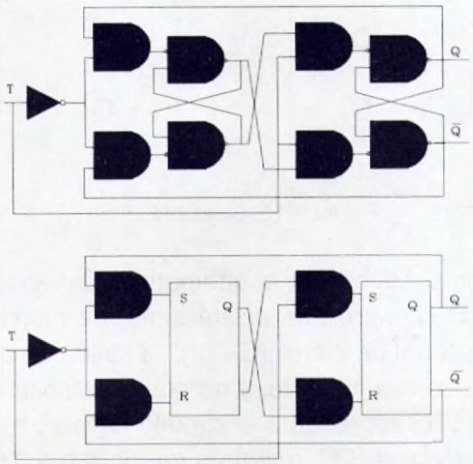


Figura 9.10. Biestable T

Su constitución se realiza a base de biestables S-R según se representa en la figura 9.10, en la que se ha empleado la configuración con puertas NAND. El conjunto de puertas lógicas del biestable S-R se simboliza mediante diagrama bloque para simplificar el gráfico.

*## **Ejemplo 9.4**: Aplicar el cronograma de la figura 9.11 a la entrada de un biestable T y obtener gráficamente la salida.*

Solución:

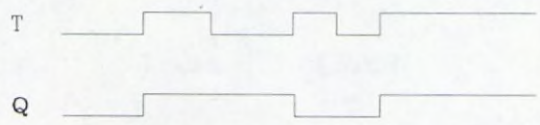


Figura 9.11

9.5. **EL BIESTABLE LATCH**

Es el biestable denominado de enclavamiento o cerrojo. Dispone de dos entradas D y C (tipo D), siendo D la entrada de datos y C la señal de cierre. El esquema bloque del mismo se indica en la figura 9.12.

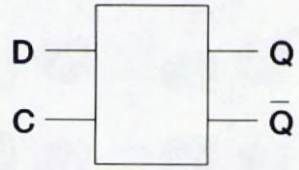


Figura 9.12. Biestable Latch

Su funcionamiento se puede esquematizar diciendo que la salida es igual a la entrada D siempre que exista señal de cierre (C="1"), y que en caso de cesar esta señal de cierre (C="0"), la salida se queda enclavada en el valor que tuviera en ese instante, y no podrá cambiar hasta que C vuelva a ser "1". La figura 9.13 representa el circuito lógico correspondiente, construido mediante biestables S-R, mientras que la tabla 9.6 corresponde a su funcionamiento.

D	C	Q	Q(+)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Tabla 9.6. Biestable Latch

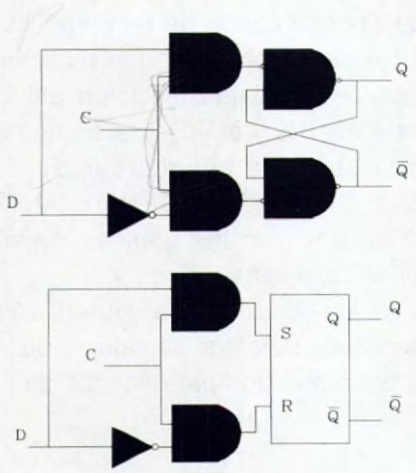


Figura 9.13. Biestable Latch

*## **Ejemplo 9.5**: Aplicar el cronograma de la figura 9.14 a la entrada de un biestable Latch y obtener gráficamente la salida.*

Solución:

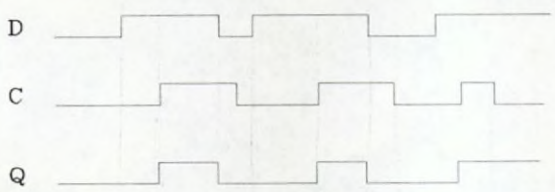


Figura 9.14

9.6. **LA SEÑAL DE RELOJ**

Al analizar la respuesta de un circuito a una señal de reloj, se ve que el sistema puede responder de maneras diferentes según que se activen las señales al comienzo del pulso del reloj o al final del mismo. En los biestables asíncronos y en los anteriores circuitos combinacionales, se han activado y desactivado según los valores o niveles lógicos de las señales que actuaban en los mismos. Esta forma de funcionamiento se denomina activación por niveles.

Un biestable activado por flancos es aquél que transmitirá a su salida los valores de las entradas en los instantes en que actúen los flancos de la señal de reloj, bien de subida o de bajada. Como puede darse el caso de solape de la señal de entrada con la de reloj en instantes en que una está bajando y la otra subiendo, o bien las dos realizan el mismo cambio de nivel, la señal de reloj realizará la activación del valor de señal que se mantenga estable entre el 10% y el 90% del tiempo de duración del flanco. Si en este intervalo la señal de entrada cambia de valor, se tendrá en cuenta EL VALOR QUE TENIA UN INSTANTE ANTES DEL CAMBIO. (En las figuras 2.2 y 2.3 se analizan las gráficas digitales y en la 2.4.(b) se introduce la señal periódica o señal de reloj).

En los gráficos que se muestran en la figura 9.15 se expresan las diversas opciones que pueden darse entre los valores de la señal de entrada y la de reloj, analizados para activación para flancos de subida y de bajada.

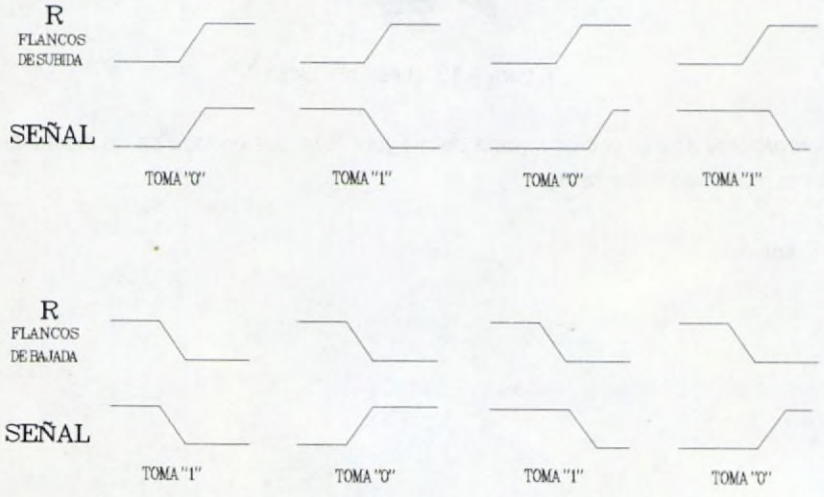


Figura 9.15. Análisis de flancos

El circuito temporizador 555 es el integrado más empleado para generar señales periódicas de reloj. Polarizado con una red RC permite obtener un amplio abanico de frecuencias en su salida. Igualmente con amplificadores operacionales se consiguen idénticos resultados con una gran estabilidad.

9.7. **EL BIESTABLE D**

Las condiciones de funcionamiento de este biestable, son similares a las del biestable de enclavamiento LATCH, pero se introduce un aspecto importante en el estudio de los biestables con la aparición de la SEÑAL DE RELOJ o CLOCK. Esta señal será la que sincronizará el funcionamiento del biestable y la que efectuará los cambios en la salida.

En concreto, el biestable D, que tendrá una entrada de datos D y una señal de reloj C, se esquematiza diciendo que la salida será igual que la entrada durante los pulsos del reloj y se mantendrá el valor de la salida cuando cese el pulso. El esquema bloque de la figura 9.16 asemeja al biestable asíncrono Latch. Si se activa por flancos de subida, el valor que exista en la entrada D cuando aparezca el flanco de subida de la señal de reloj es el que se transmite a la salida y se mantiene hasta el siguiente flanco de subida de la señal de reloj. Igualmente ocurre con el activado por flancos de bajada, particularizando para este flanco. Tal como se indicó en la señal de reloj, si los flancos coinciden se toma el valor que tuviera la entrada de datos un instante antes.

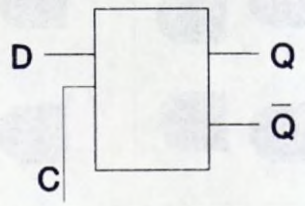


Figura 9.16. Biestable D

Si se analiza un diagrama de tiempos o cronograma de las señales que actúan en el biestable síncrono D, podemos obtener la salida gráficamente según se indica en la figura 9.17. En ella, se ha supuesto una secuencia cualquiera de señales en la entrada D, con período múltiple del correspondiente a la señal de sincronía de reloj.

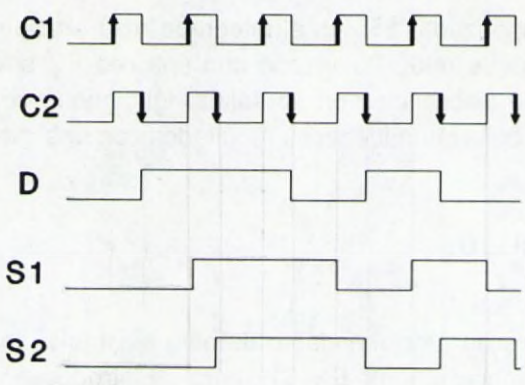


Figura 9.17. Cronograma del biestable D

Observando con detalle la figura 9.17, se observa que al estar sincronizada la señal de reloj con la entrada al circuito D, el resultado obtenido a la salida es equivalente tanto si se realiza la activación con flancos de subida (S_1) como de bajada (S_2), existiendo entre ambos resultados la única diferencia de los instantes de tiempo en que aparecerán las señales de salida.

Al mismo tiempo, se puede comprobar que ambas señales de salidas son idénticas a la que aparece a la entrada D, pero desfasadas un período de tiempo equivalente a un ciclo completo de la señal de reloj para el caso de activación por flancos de bajada y equivalente a un ciclo completo menos la

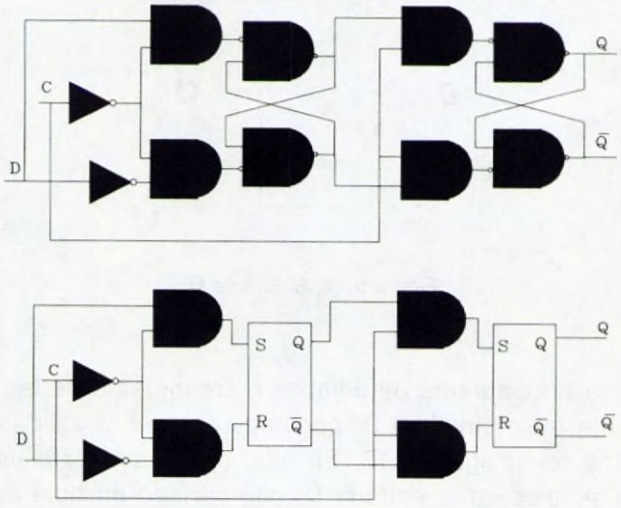


Figura 9.18. Biestable D

duración del pulso de la señal del reloj comprendida entre los flancos de subida y bajada para el caso de utilizar activación por flanco de subida. Por tanto, se puede establecer que el biestable D desfasa la señal que se introduce en la entrada.

El diagrama de puertas lógicas de este biestable se representa en la figura 9.18, donde se han realizado dos versiones del mismo, la primera con puertas lógicas únicamente y la segunda empleando bloques representativos de los biestables S-R. El circuito de la figura es un modelo muy utilizado, y recibe la denominación MAESTRO-ESCLAVO o MASTER-SLAVE. Se debe esta denominación a la conexión de los biestables, de manera que el que está representado como biestable 1 recibe señal del biestable 2 pero no viceversa.

Esto se observa analizando el circuito y deduciendo que en el primer biestable (el primero que recibe la señal externa), que se ha definido como biestable 2, la señal presente como estado interno en la salida Q se queda enclavada en los períodos de tiempo en que C="0", es decir, cuando no existe señal de reloj. Durante el ciclo de activación de la señal de reloj (C="1"), la señal que se mantenía enclavada en Q_2 pasa al segundo biestable definido como biestable 1. Al mismo tiempo, el primer biestable (biestable 2) está bloqueado y por tanto Q_2 no puede tomar ningún otro valor hasta que cambie la señal de reloj C. Su tabla de funcionamiento es la siguiente:

Q	Q(+)	D
0	0	0
0	1	1
1	0	0
1	1	1

Tabla 9.7. Biestable D

*## **Ejemplo 9.6**: Aplicar el cronograma de la figura 9.19 a la entrada de un biestable D y obtener gráficamente la salida empleando activación por flancos de subida.*

Solución:

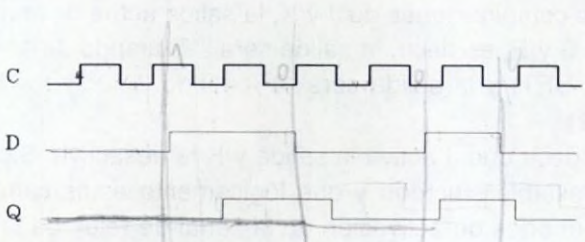


Figura 9.19

*## **Ejemplo 9.7**: Aplicar el cronograma de la figura 9.20 a la entrada de un biestable D y obtener gráficamente la salida empleando activación por flancos de bajada.*

Solución:

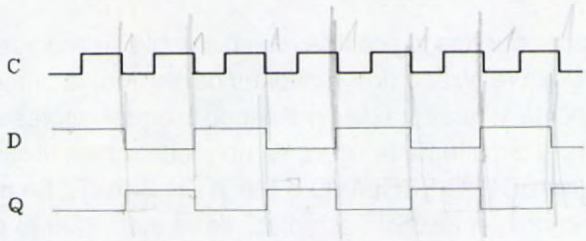


Figura 9.20

9.8. **EL BIESTABLE J-K**

El biestable síncrono J-K es uno de los biestables más utilizado y es tal vez el más representativo de los empleados en diseño de circuitos secuenciales. Su concepción de funcionamiento es igual a la del biestable asíncrono S-R, pero salvando el problema de inespecificación general de la salida para la entrada "11". Su esquema bloque se representa en la figura 9.21.

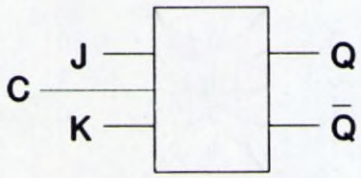


Figura 9.21. Biestable J-K

En el biestable J-K si las dos entradas son "1" la salida cambia de estado. Para las otras combinaciones de J y K, la salida actúa de la misma forma que lo hacía para S y R, es decir, la salida será "1" cuando J="1" y K="0" (equivale a la orden SET) y la salida será "0" cuando J="0" y K="1" (equivale a la orden RESET).

Se puede decir que J activa la salida y K la desactiva. Siempre sin olvidar que es un biestable síncrono y que lógicamente estos cambios se realizan durante los períodos de activación de la señal de reloj. La tabla de funcionamiento del biestable puede representarse de la siguiente forma:

J	K	Q	Q(t)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Tabla 9.8. Biestable J-K

También en este circuito hay 2 biestables S-R que forman una estructura MASTER-SLAVE o MAESTRO-ESCLAVO. Su funcionamiento es algo diferente a lo analizado en el caso del biestable D, ya que es necesario introducir modificaciones para evitar señales prohibidas o parásitas. Estas pueden ocurrir durante el pulso de reloj, y son pequeños pulsos de duración inferior al de reloj, que pueden aparecer en J y en K mientras la señal de reloj C es "1". De esta manera, los circuitos reales se fabrican de tal forma que anulan cualquier efecto que estas pequeñas señales anómalas puedan tener en la salida de los mismos. Por tanto, el funcionamiento del biestable maestro-esclavo

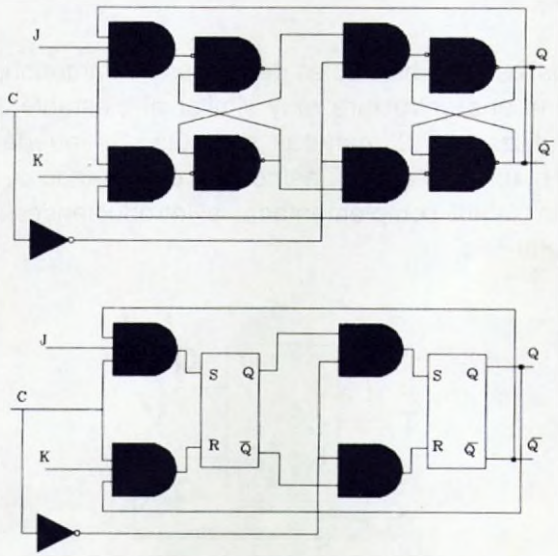


Figura 9.22. Biestable J-K

dependerá de los flancos de la señal de reloj y en este caso, de los 2 flancos, tanto del de subida como del de bajada. En general se puede decir que los biestables J-K funcionan CAPTURANDO EL DATO DE LAS ENTRADAS EN EL FLANCO DE SUBIDA DE LA SEÑAL DE RELOJ y TRANSMITIENDO DICHO VALOR A LA SALIDA EN EL FLANCO DE BAJADA. La señal de salida se mantiene hasta el siguiente flanco de bajada de la señal de reloj. (Existen algunos modelos de biestables J-K con estructura y funcionamiento algo diferente, pero su uso es muy reducido).

Si la señal de las entradas hace cambiar el estado de la salida durante el flanco de subida de la señal de reloj, el primer biestable registra la nueva expresión de la salida y no la transmite al segundo biestable hasta que el flanco de bajada de la señal de reloj active el circuito. Si la señal de las entradas cesa antes del flanco de bajada de la señal de reloj, no afectaría a la salida ya que el primer biestable no se modifica al quedarse enclavado. La tabla resumen del funcionamiento del biestable J-K es la 9.8.

Q	Q(t)	J	K
0	0	0	-
0	1	1	-
1	0	-	1
1	1	-	0

Tabla 9.8. Biestable J-K

Si analizamos los otros biestables detallados con anterioridad, el biestable T asíncrono tiene una estructura muy similar al biestable J-K, pudiéndose representar y utilizar de 2 maneras distintas. Se puede transformar el biestable J-K en un biestable T asíncrono conectando K a la salida del biestable, J a la salida complementada, e introduciendo la señal T en la correspondiente al reloj.

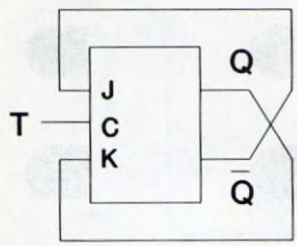


Figura 9.23. Biestable T asíncrono

La segunda opción sería transformar el biestable J-K en un biestable T síncrono. Para ello, se mantendrá la señal de reloj C y se conectarán las entradas J y K a la entrada de señal T. El esquema de conexiones de este biestable T a partir del J-K se representa en la figura 9.24.

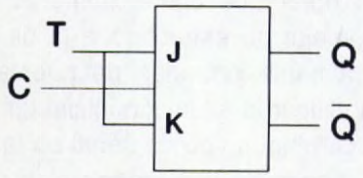


Figura 9.24. Biestable T síncrono

*## **Ejemplo 9.8**: Aplicar el cronograma de la figura 9.25 a la entrada de un biestable J-K y obtener gráficamente la salida.*

Solución:

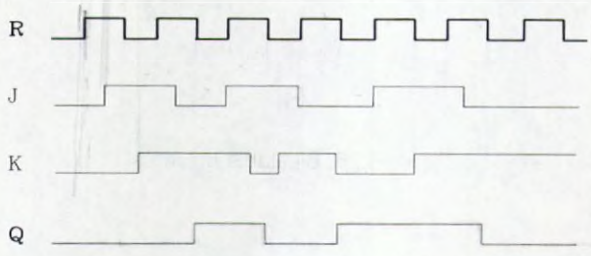


Figura 9.25

9.9. EL BIESTABLE UNIVERSAL

Se denomina biestable universal a un biestable síncrono-asíncrono, que se forma por la unión de las funciones desarrolladas por los biestables S-R y J-K. Siendo su esquema de representación el de la figura 9.26, puede funcionar en modo síncrono o asíncrono según las entradas que se utilicen y en función de la aplicación o no de la señal de reloj.

Como se ha desarrollado en los apartados anteriores, los biestables T, LATCH, D, se pueden expresar en función de los biestables S-R y J-K, según que se trate de diseñar un circuito síncrono o asíncrono respectivamente. La unión de ambos en un sólo biestable permite la inmediata conversión de un

circuito síncrono en asíncrono y viceversa. Ello conlleva aparejada la consiguiente ventaja que supone en el aspecto económico y en el tiempo de diseño y montaje de circuitos, especialmente cuando no se han delimitado los parámetros del circuito a diseñar y se trata de obtener resultados que conduzcan al diseño óptimo del mismo.

Es interesante hacer notar que comercialmente la mayor parte de los biestables tienden a asemejar su estructura a la de un biestable universal, dado que poseen normalmente entradas de puesta a cero y carga. Son señales que permiten modificar la situación inicial de respuesta del biestable sin necesidad de introducir ningún tipo de señal en las entradas de datos del mismo. Su denominación coincide con las iniciales del biestable S-R, set o preset corresponde con la carga inicial y reset con la puesta a cero.

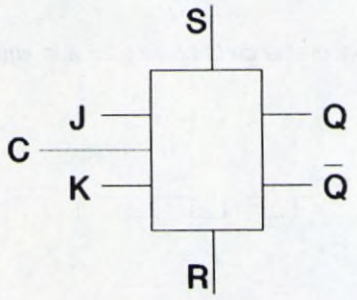


Figura 9.26. Biestable universal

9.10. **EL DISPARADOR SCHMITT**

El DISPARADOR SCHMITT o SCHMITT TRIGGER es un circuito que transforma señales oscilantes en ondas de perfil perfectamente definido, por lo que se emplea para fijar niveles de tensión y detectar los puntos de superación de dichas referencias. Son muy empleados en generación de señales de reloj. Están basados en el uso de técnicas de histéresis, lo que produce incluso su introducción en el esquema representativo de los mismos. Normalmente el disparador va asociado con las puertas lógicas convencionales, encontrándose así inversores con disparador Schmitt, puertas NAND, AND, etc.

La función de transferencia o ciclo de histéresis es la relación entre las tensiones de entrada (V_i) y de salida (V_o). Para determinar su funcionamiento, se establecen dos tensiones de umbral V_{T-} y V_{T+} que aplicados a la entrada del circuito marcan los puntos de disparo. Analizando por ejemplo la gráfica 9.29 correspondiente al inversor con disparador Schmitt 7414, el fabricante

delimita para V_{T-} y V_{T+} los valores típicos 0,8^{V} y 1,67^{V} respectivamente, que proporcionarán un valor mínimo de salida V_{oL}=0,25^{V} y un valor máximo V_{oH}=3,4^{V} con t_{PLH}=t_{PHL}=15^{ns}.

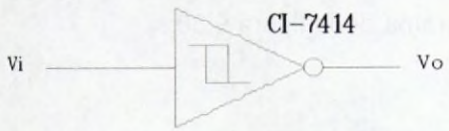


Figura 9.27. Inversor con disparador Schmitt

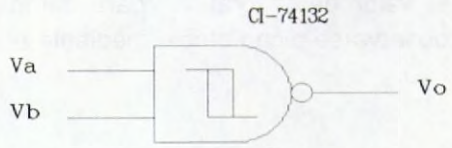


Figura 9.28. Puerta NAND con disparador Schmitt

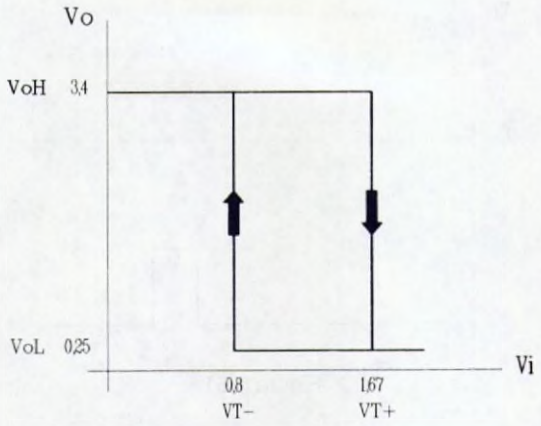


Figura 9.29. Función de transferencia de un inversor Schmitt

Al tratarse de un inversor, con 0^{V} de entrada la salida lógica será un "1", o su equivalente en tensión, 3,4^{V}. Aunque aumente la tensión de entrada, la tensión de salida seguirá invariable hasta que se alcance el valor V_{T+}=1,67^{V}, momento en el que descenderá hasta 0,25^{V}, valor mínimo de V_{o}. Si sigue

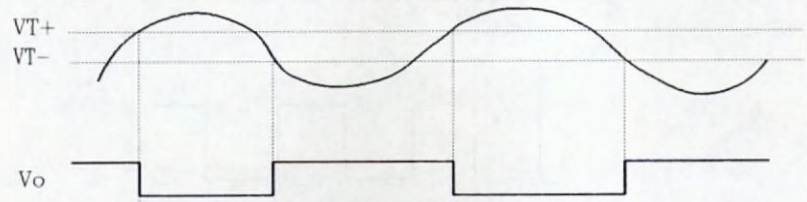


Figura 9.30. Relación entre tensiones umbral y salida

aumentando la tensión de entrada no habrá variación de la salida, situación que sucede si disminuye V_i pero no se alcanza el valor de umbral $V_{T-}=0,8^V$ que produce una nueva elevación de la salida a $3,4^V$. La progresiva disminución de la entrada no afecta a la salida, necesitándose nuevamente superar el valor de umbral V_{T+} para cambiar el valor de V_o. Gráficamente puede observarse dicho efecto mediante el diagrama de la figura 9.30.

*## **Ejemplo 9.9**: Obtener la salida del inversor Schmitt si la tensión de entrada corresponde al gráfico de la figura 9.31.*

Solución:

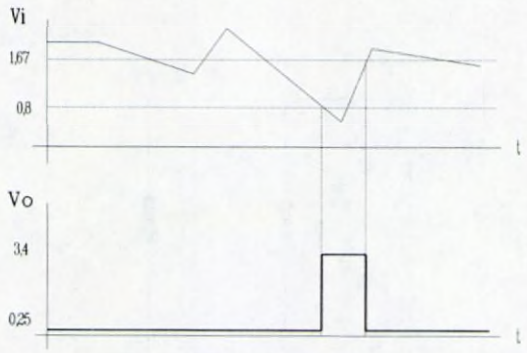


Figura 9.31

*## **Ejemplo 9.10**: Obtener la salida del inversor Schmitt si la tensión de entrada corresponde al gráfico de la figura 9.32.*

Solución:

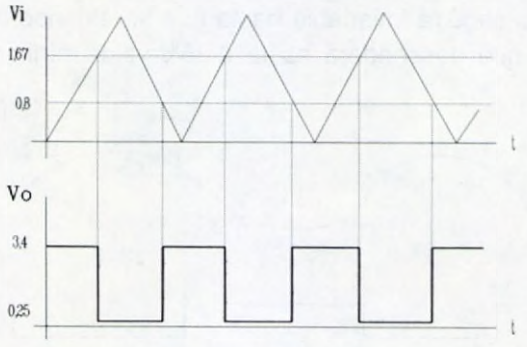
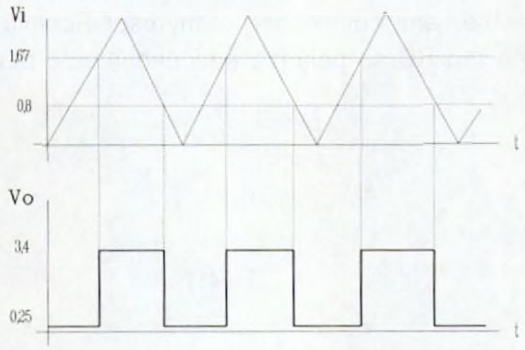


Figura 9.32

*## **Ejemplo 9.11**: Trazar la función de transferencia de un disparador Schmitt a partir de los gráficos de entrada y salida de la figura 9.33.*



Solución:

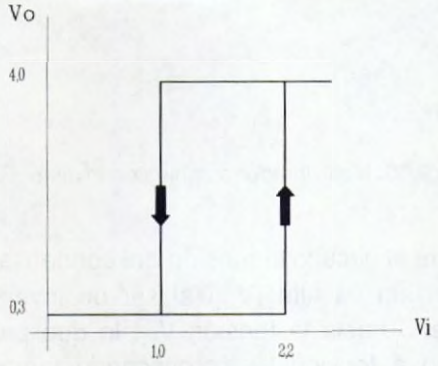


Figura 9.34

9.11. **CIRCUITOS MULTIVIBRADORES**

El multivibrador es un circuito que conmuta de valor entre "0" y "1" de forma continuada, bien autónomamente o mediante disparo exterior. Existen treş grupos básicos de multivibradores, denominados astable, monoestable y biestable.

El multivibrador astable genera un tren de pulsos de manera independiente, sin disparo exterior, pudiendo seleccionarse la frecuencia y el ancho del pulso según los distintos circuitos. El multivibrador monoestable genera un pulso cuando es disparado externamente, lo que implica la necesidad de un disparo cíclico si se desea obtener una señal periódica o repetitiva. El multivibrador biestable o flip-flop en sus distintas vertientes ha sido tratado a lo largo del presente capítulo.

9.11.1. El disparador Schmitt como multivibrador astable

Uno de los circuitos más sencillos que se pueden construir como multivibrador astable tiene como base el inversor Schmitt 7414 o similares. Con una pequeña red RC se polariza adecuadamente para generar un tren de pulsos.

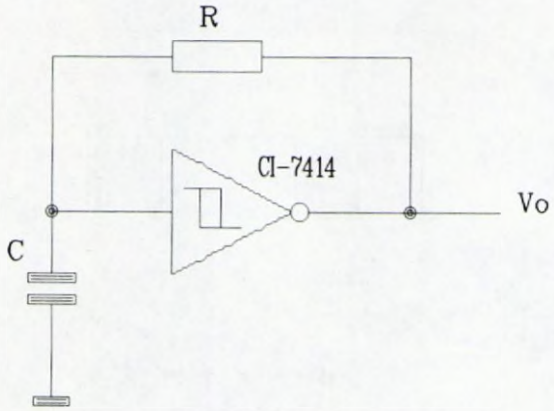


Figura 9.35. Multivibrador astable con inversor Schmitt

Al poner en marcha el circuito la tensión del condensador C es nula, por lo que la salida del circuito es alta (V_{oH}) al ser un inversor. Inmediatamente comienza a cargarse C hasta la tensión V_{T+}, lo que provoca el disparo del inversor y su cambio a tensión V_{oL}, provocando también la descarga del condensador. Cuando la tensión de C llegue al valor V_{T-} el inversor volverá a dispararse alcanzando la salida nuevamente el valor V_{oH}. Este ciclo de trabajo se repite continuamente generándose un tren de pulsos.

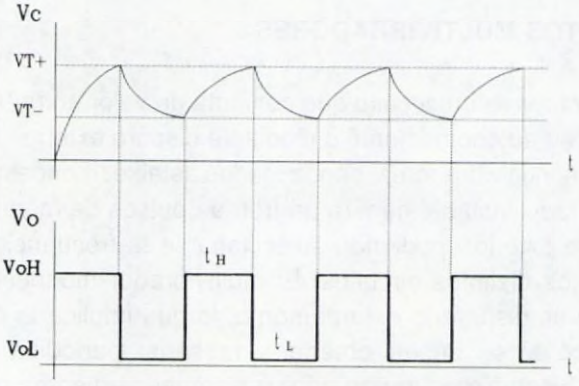


Figura 9.36. Tren de pulsos del multivibrador astable con disparador Schmitt

Denominando t_H y t_L respectivamente al ancho de las zonas "1" y "0" del tren de pulsos una vez estabilizada la señal (después del segundo ciclo de carga y descarga de C), se obtienen las siguientes ecuaciones:

[9.2] $$t_H = RC\ln\frac{(V_{oH}-V_{T-})}{(V_{oH}-V_{T+})}$$

[9.3] $$t_L = RC\ln\frac{(V_{oL}-V_{T+})}{(V_{oL}-V_{T-})}$$

[9.4] $$T = t_H + t_L$$

La elección del circuito inversor así como los valores de resistencia y capacidad determinan el tipo de onda que se obtiene en la salida, pudiendo obtenerse un abanico de frecuencias bastante amplio.

***## Ejemplo 9.12**: Trazar las ondas de carga y descarga del condensador y el tren de pulsos de un multivibrador astable con inversor Schmitt 7414, cuyas características son $V_{oH}=5^V$, $V_{oL}=0^V$, $V_{T+}=1,67^V$, $V_{T-}=0,8^V$, $t_{PLH}=15^{ns}$, $t_{PHL}=15^{ns}$.*

Solución:

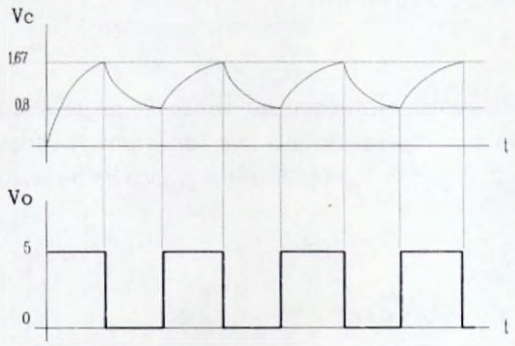


Figura 9.37

***## Ejemplo 9.13**: Calcular t_H, t_L, T, f para el multivibrador del ejemplo 9.11 si $R=20^K$, $C=0,015^F$.*

Solución:

$t_H=20*10^3*15*10^{-9}*\ln((5-0,8)/(5-1,67))=69,63*10^{-6}=69,63^s$

$t_L=20*10^3*15*10^{-9}*\ln((0-1,67)/(0-0,8))=220,79*10^{-6}=220,79^s$

$T=t_H+t_L=290,42^s$

$f=1/290,42*10^{-6}=3443,29^{Hz}=3,44^{KHz}$

*## **Ejemplo 9.14**: Calcular la frecuencia de la señal peródica obtenida con un multivibrador astable con inversor Schmitt cuyas ondas se representan en la figura 9.38, para valores de $R=15^K$ y $C=10^F$.*

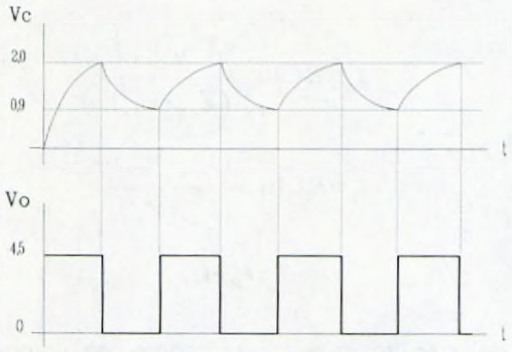


Figura 9.38

Solución:

De la figura se obtiene $V_{T-}=0,9^V$ $V_{T+}=2,0^V$ $V_{oH}=4,5^V$ $V_{oL}=0^V$

$t_H=15*10^3*10*10^{-6}*ln((4,5-0,9)/(4,5-2))=54,7*10^{-3}=54,7^{ms}$

$t_L=15*10^3*10*10^{-6}*ln((0-2)/(0-0,9))=15,8*10^{-3}=15,8^{ms}$

$T=54,7*10^{-3}+15,8*10^{-3}=70,5^{ms}$

$f=1/T=1/70,5*10^{-3}=14,18^{Hz}$

*## **Ejemplo 9.15**: Calcular el margen de variación de la frecuencia de un circuito multivibrador astable con inversor Schmitt que posee una R variable entre 1 y 10^K, con $C=1^F$, $V_{T-}=1,5^V$, $V_{T+}=3^V$, $V_{oH}=5^V$ y $V_{oL}=0^V$. Trazar el gráfico de histéresis.*

Solución:

$t_{H1}=10^3*10^{-6}*ln((5-1,5)/(5-3))=0,56^{ms}$

$t_{L1}=10^3*10^{-6}*ln((0-3)/(0-1,5))=0,693^{ms}$

$T_1=1,253^{ms}$

$f_1=1/T_1=798^{Hz}$

$t_{H2}=10*10^3*10^{-6}*ln((5-1,5)/(5-3))=5,6^{ms}$

$t_{L2}=10*10^3*10^{-6}*ln((0-3)/(0-1,5))=6,93^{ms}$

$T_2=12,53^{ms}$

$f_2=1/T2_1=79,8^{Hz}$

El margen de frecuencias es $79,8^{Hz}<f<798^{Hz}$

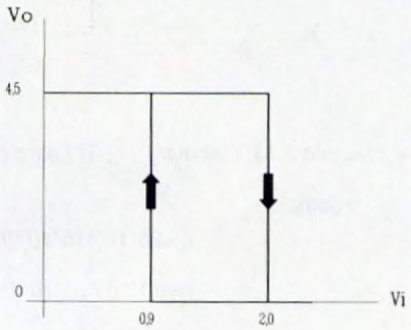


Figura 9.39

9.11.2. **El 555 como multivibrador astable**

El circuito integrado 555 es uno de los más conocidos y empleados generadores de señales de reloj. Se emplea para múltiples usos, destacando la generación de señales como multivibrador astable y como multivibrador monoestable.

Su estructura interna está formada básicamente por un biestable S-R y dos comparadores analógicos, fabricándose como DIL de 8 terminales. Para que funcione como astable es necesario conectar externamente una red como la indicada en la figura 9.40.

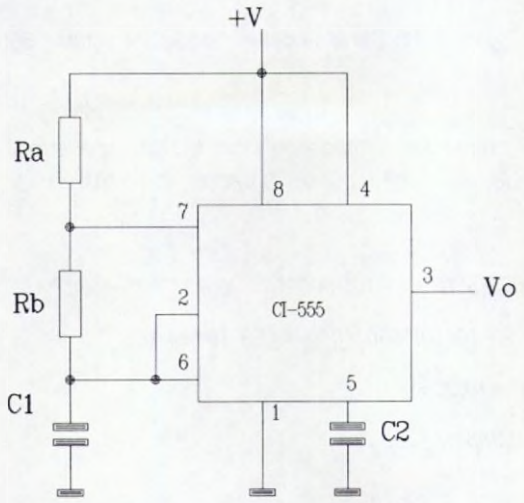


Figura 9.40. Multivibrador astable 555

El efecto de carga y descarga del condensador C_1 hace que la tensión en sus extremos V_{C1} sea oscilante entre +V/3 y +2V/3 siendo +V la tensión de alimentación, que normalmente oscila entre 4,5 y 18^V según las variantes. Los fabricantes determinan unas ecuaciones para obtención de los tiempos de ciclo en función de la red externa de polarización. De esta forma se obtienen:

$$t_L = 0{,}693 R_b C_1 \qquad [9.5]$$

$$t_H = 0{,}693 (R_a + R_b) C_1 \qquad [9.6]$$

$$V_{oH} = V - 1{,}5 \qquad [9.7]$$

$$V_{oL} = 0{,}1 \qquad [9.8]$$

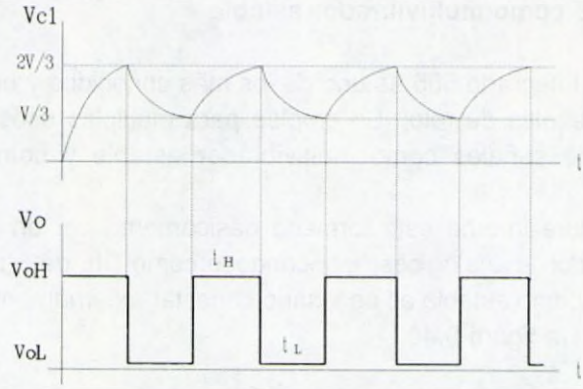


Figura 9.41. Señales del multivibrador astable 555

*## **Ejemplo 9.16**: Obtener los tiempos de ciclo t_H, t_L, el período y la frecuencia de un astable 555 polarizado con $R_a=5^K$, $R_b=10^K$, $C_1=600^{pF}$, $C_2=0,01^F$ y $V=5^V$.*

Solución:

$t_H=0,693(5*10^3+10*10^3)600*10^{-12}=6,237*10^{-6}=6,237^s$

$t_L=0,693*10*10^3*600*10^{-12}=4,158*10^{-6}=4,158^s$

$T=t_H+t_L=10,395^s$

$f=1/10,395*10^{-6}=96,2^{KHz}$

*## **Ejemplo 9.17**: En el circuito descrito en el ejemplo 9.16 se sustituye R_b por una resistencia variable entre 1 y 10^K. Obtener el margen de frecuencia que puede proporcionar el circuito.*

Solución:

$t_{H1}=0,693(5*10^3+10^3)600*10^{-12}=2,49^s$

$t_{L1}=0,693*10^3*600*10^{-12}=0,42^s$

$T_1=2,91^s$

$f_1=1/T_1=0,344^{MHz}$

Los cálculos para $R_b=10^K$ están realizados en el problema 9.16 y proporcionan una frecuencia:

$f_2=96,2^{Khz}$

El margen de variación de la frecuencia es $96,2^{KHz}<f<344^{KHz}$

9.11.3. **El 555 como multivibrador monoestable**

Como se indicaba en el apartado anterior el circuito integrado 555 puede utilizarse para generar pulsos mediante disparo exterior. Igual que en el astable, este multivibrador necesita una red exterior de polarización como se indica en la figura 9.42.

Cuando se produce un pulso (negativo) de disparo, de muy corta duración, aplicado al terminal V_t, el condensador C_1 comienza a cargarse hasta +2V/3, momento en el que se descarga de manera rapidísima. El tiempo que dura la carga del condensador es el ancho del pulso generado en la salida V_o según se aprecia en las ondas del circuito. El período de la señal de salida es el mismo que el de la señal de disparo, pudiendo por tanto obtenerse un sólo pulso en la salida o una señal periódica en función del tipo de señal de disparo que se aplique. Las tensiones V_{oH} y V_{oL} son iguales que en el 555 astable mientras que t_H se obtiene con la siguiente ecuación:

$$t_H = 1{,}1 R_a C_1 \qquad [9.9]$$

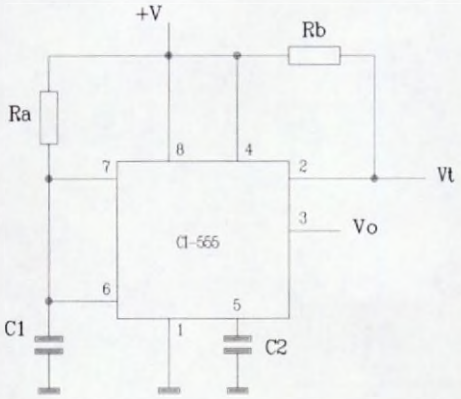


Figura 9.42. Multivibrador monoestable 555

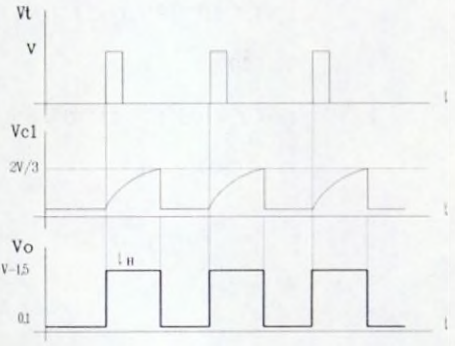


Figura 9.43. Señales en el monoestable 555

*## **Ejemplo 9.18**: Obtener los tiempos de ciclo t_H, t_L, el período y la frecuencia de un astable 555 polarizado con $R_a=5^K$, $R_b=10^K$, $C_1=600^{pF}$, $C_2=0{,}01^F$ y $V=5^V$. La tensión de disparo es de 5^V en forma de pulso negativo de duración 2^s repitiéndose cada 100^s.*

Solución:

$t_H=1,1*5*10^3*600*10^{-12}=3,3^s$

$T=100*10^{-6}=100^s$

$t_L=T-t_H=96,7^s$

$f=1/10^{-4}=10^{KHz}$

*## **Ejemplo 9.19**: En el circuito descrito en el ejemplo 9.18 se sustituye R_a por una resistencia variable entre 1 y 10^K. Obtener el margen de frecuencia que puede proporcionar el circuito.*

Solución:

Se obtiene la misma frecuencia de salida, ya que ésta sólo depende de la frecuencia de la señal de disparo V_t y no del circuito de polarización.

*## **Ejemplo 9.20**: Representar las ondas de disparo, carga de C_1 y salida de un multivibrador monoestable 555 polarizado con $R_a=10^K$, $R_b=10^K$, $C_1=1^{nF}$, $C_2=0,1^F$ y $V=10^V$. La tensión de disparo es de 10^V en forma de pulso negativo de duración 1^{ns} repitiéndose cada 100^s.*

Solución:

$t_H=1,1*10*10^3*10^{-9}=11*10^{-6}=11^s$

$T=100^s$

$t_L=T-t_H=100*10^{-6}-11*10^{-6}=89^s$

$f=1/100*10^{-6}=10^{KHz}$

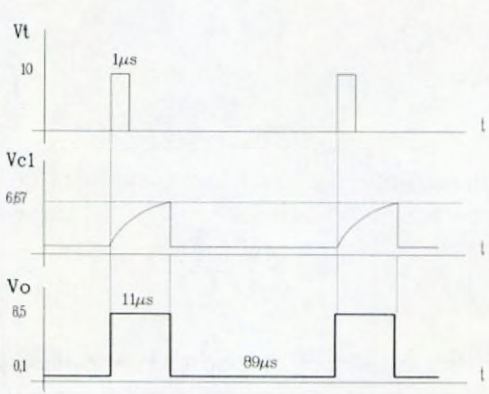


Figura 9.44

9.11.4. **Multivibrador monoestable 74121**

Un circuito también bastante utilizado para generar pulsos mediante disparo exterior es el integrado 74121. Contiene un disparador Schmitt en su interior junto con un biestable de conmutación T. La sucesiva aplicación de señales de disparo en su entrada proporciona una señal periódica en su salida.

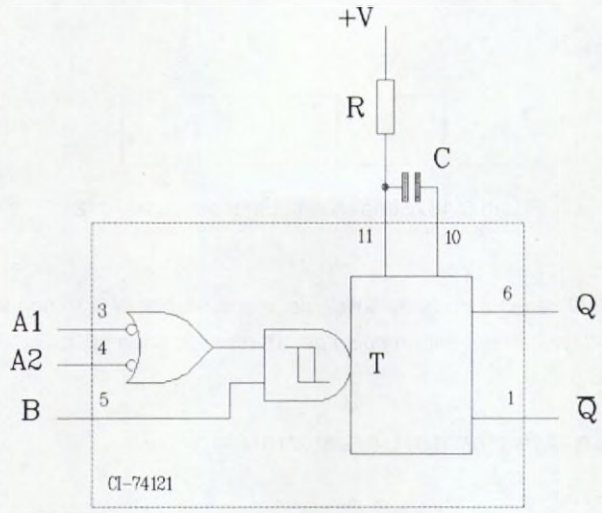


Figura 9.45. Multivibrador monoestable 74121

Este circuito requiere una red externa de polarización muy pequeña y sencilla en comparación con otros osciladores. Las señales externas A_1, A_2 y B proporcinan los niveles de disparo necesarios en la puerta AND con disparador Schmitt. Con B en valor "1" cualquier "0" en las entradas A disparará el circuito mientras que con las entradas A fijadas en "0" (al menos una de ellas) cualquier valor "1" en B producirá el disparo.

El terminal 9 de dicho integrado proporciona una resistencia interna de 2^K pero normalmente no es utilizada al ser sustituida por una resistencia externa. El ancho de duración del pulso se obtiene de la siguiente ecuación:

$$t_H = 0{,}693RC \qquad [9.10]$$

El período de la onda de salida es el mismo que proporciona la señal de disparo y los niveles de tensión de salida al corresponder con terminales de un biestable son los niveles lógicos "0" y "1" cuyos valores dependen del

integrado y de la lógica de fabricación usual en circuitería digital. Si por ejemplo se conectan a "0" los terminales A_1 y A_2, la introducción de pulsos repetitivos en B proporciona la salida de la figura 9.46.

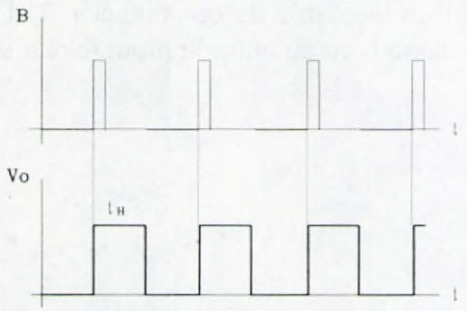


Figura 9.46. Señales en el monoestable 74121

*## **Ejemplo 9.20**: Obtener la onda de salida del monoestable 74121 cuando A_1=0, A_2=1 y B es un señal periódica formada por un pulso de 10^{s} repetido cada 100^{s}. $R=10^{K}$, $C=0,01^{F}$ $V=5^{V}$.*

Solución:

$t_H=0,693*10*10^{3}*0,01*10^{-6}=69,3*10^{-6}=69,3^{s}$

$T=100^{s}$

$t_L=T-t_H=30,7^{s}$

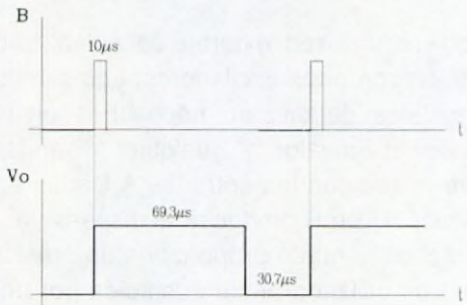


Figura 9.47

*## **Ejemplo 9.21**: Calcular el valor de la resistencia externa que se debería colocar en el circuito del problema 9.20 para obtener una señal de $t_H=50^{s}$.*

Solución:

$R=t_H/0,695C=50*10^{-6}/(0,695*10^{-9})=71,94^{K}$

9.11.5. **Multivibrador astable 74124 controlado por cristal de cuarzo**

Los multivibradores polarizados externamente con redes RC generan buenas señales de reloj pero no son extremadamente estables como requieren algunos circuitos digitales. Como pueden tener lugar variaciones en los parámetros del circuito por efecto de la temperatura, ruidos y desajustes a lo largo del tiempo, se utiliza un cristal de cuarzo para proporcionar dicha estabilidad.

El cristal de cuarzo vibra a una determinada frecuencia según el proceso de corte a que haya sido sometido, proporcionando una señal casi inalterable a las condiciones de variación que sí afectan a los componentes pasivos. Este elemento se incorpora normalmente a los circuitos generadores de señal de reloj haciendo las veces del condensador de polarización externo.

Un circuito muy empleado como multivibrador es el integrado 74124, doble oscilador controlado por tensión externa con conexión para cristal de cuarzo, que proporciona un espectro de salida entre 1^{Hz} y 60^{MHz}.

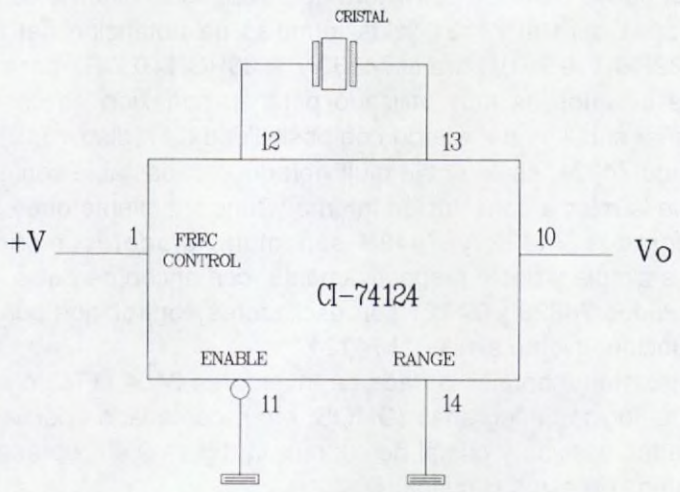


Figura 9.48. Oscilador 74124

La frecuencia de salida se puede variar mediante los terminales de control de frecuencia (control frequency) y de margen de frecuencia (range frequency), normalmente activados alrededor de 5^{V}, mientras que el terminal "Enable" actúa como en todos los integrados, bloqueando o desbloqueando

el funcionamiento del mismo. Siendo f_o la frecuencia de salida y C_{ext} la capacidad externa medida entre los terminales donde se conecta el cristal de cuarzo o en su defecto un condensador:

$$f_o = \frac{5*10^{-4}}{C_{ext}} \qquad [9.11]$$

9.11.6. **Otros circuitos osciladores**

Existen en el mercado un abanico bastante amplio de circuitos osciladores autónomos o controlados externamente con redes pasivas o cristal de cuarzo con numerosas aplicaciones en diseño de circuitos digitales. En los catálogos de circuitos integrados pueden encontrarse circuitos como el 74122 y 74123, multivibradores monoestables redisparables, de constitución interna muy similar al 74121 pero con la posibilidad de ser redisparados sin necesidad de llevar el disparador Schmitt al valor correspondiente de conmutación para el comienzo del nuevo ciclo. Su estructura de polarización externa es muy similar a la utilizada para el 74121 y las fórmulas de obtención del ancho del pulso es 0,32RC(1+0,7/R) para el 74132 y 0,28RC(1+0,7/R) para el 74123. Este tipo de circuitos es muy utilizado para su conexión en cascada que permite diseñar circuitos de retardo con posibilidad de redisparo.

El integrado 74221, es un doble multivibrador monoestable con disparador Schmitt, tiene la misma constitución interna y funcionamiento que el 74121.

Los integrados 74422 y 74423 son multivibradores monoestables redisparables simple y doble respectivamente, con ancho de pulso 0,33RC.

Los integrados 74320 y 74321 son osciladores controlados por cristal de cuarzo de funcionamiento similar al 74124.

También existen montajes a base de inversores 7404 (TTL) o su equivalente en otras lógicas integradas (CMOS, etc.), controlados por las dos variantes de redes pasivas y cristal de cuarzo. La figura 9.49 representa como ejemplo algunos de estos circuitos.

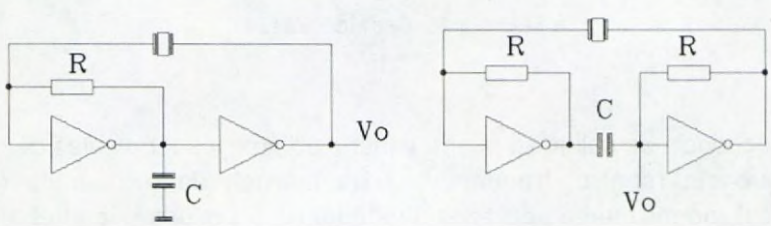


Figura 9.49. Osciladores con inversores

9.12. EJEMPLOS

***## Ejemplo 9.22**: Obtener gráficamente la salida de cada biestable del circuito de la figura 9.50 cuando se aplican las entradas del cronograma de la figura 9.51. (Biestables T, S-R de inscripción prioritaria, D activado por flancos de subida y J-K).*

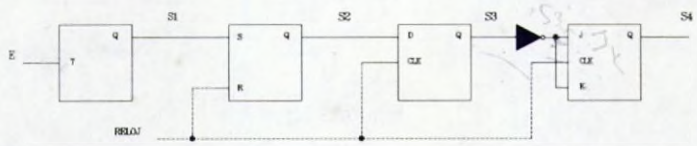


Figura 9.50

Solución:

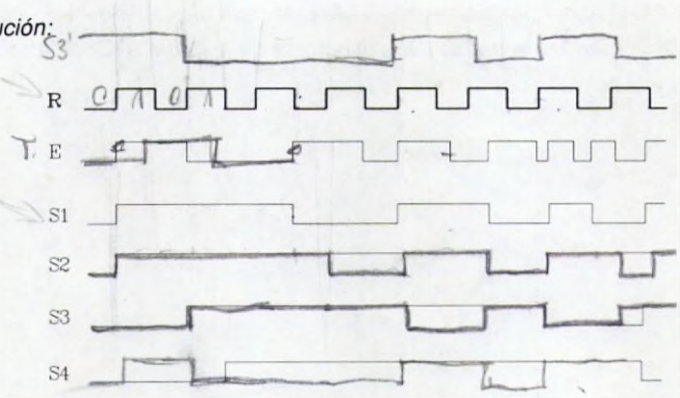


Figura 9.51

***## Ejemplo 9.23**: Obtener gráficamente la salida de cada biestable del circuito de la figura 9.52 cuando se aplican las entradas del cronograma de la figura 9.53.*

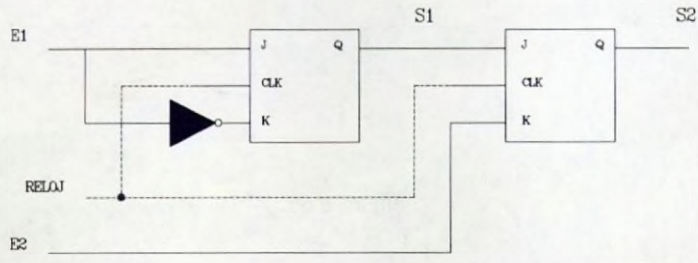


Figura 9.52

Solución:

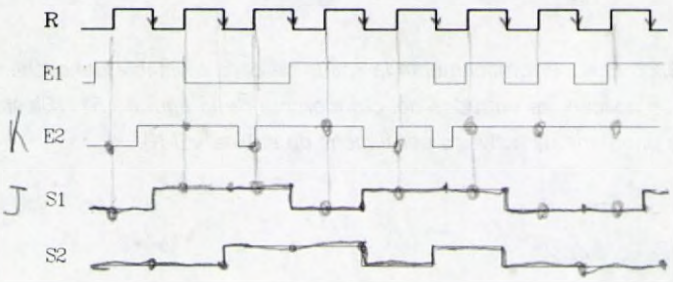


Figura 9.53

***## Ejemplo 9.24**: Obtener gráficamente la salida de cada biestable del circuito de la figura 9.54 cuando se aplican las entradas del cronograma de la figura 9.55. (T asíncrono formado con un J-K y D activado por flancos de bajada).*

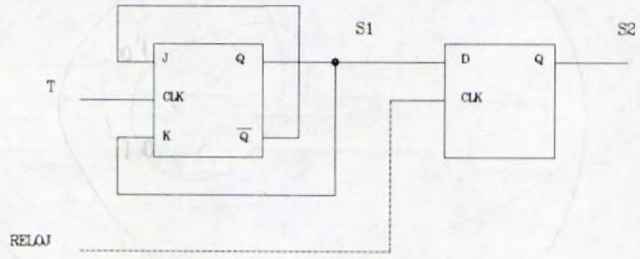


Figura 9.54

Solución:

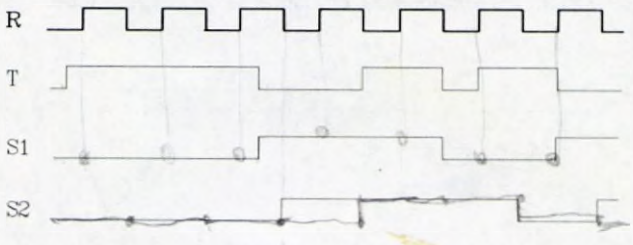


Figura 9.55

*## **Ejemplo 9.25**: Obtener gráficamente la salida de cada biestable del circuito de la figura 9.56 cuando se aplican las entradas del cronograma de la figura 9.57. (Biestables T, J-K, D activado por flancos de subida, S-R de borrado prioritario y Latch).*

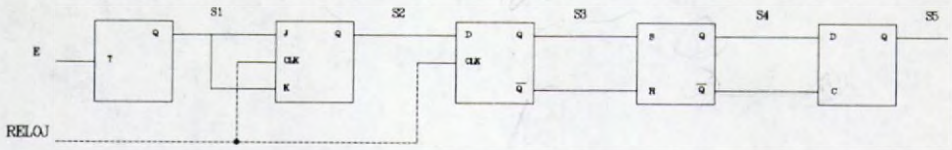


Figura 9.56

Solución:

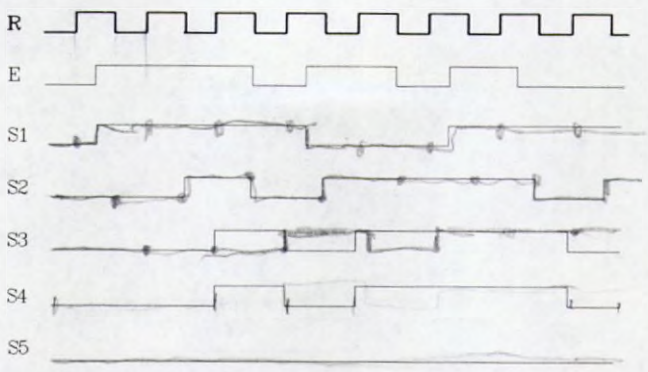


Figura 9.57

TEMA 10
CONTADORES

10

CONTADORES

10.1. INTRODUCCION

En el tema que se describe a continuación, se analizan en profundidad los distintos tipos de contadores existentes. Se comienza por un estudio completo del contador síncrono, el más importante, describiendo distintas posibilidades en cuanto al número de bits a emplear.

Dentro de los contadores síncronos analizados secuencialmente, se incluyen el contador síncrono binario, el contador síncrono BCD y el contador síncrono BCD Exceso-3. Posteriormente se analizan otros contadores síncronos como el de anillo y el conmutado en cola de Johnson, que son analizados partiendo de su circuito de biestables.

Por último, se analiza una versión muy conocida y utilizada de contador asíncrono, proveniente de una modificación de contador síncrono, siendo un esquema genérico que puede servir de modelo para la formación de otros contadores binarios asíncronos. Todos los contadores incluyen tablas de funcionamiento, cronogramas y conexionado externo, que facilite en todo momento el completo conocimiento de los mismos y de sus etapas.

10.2. EL CONTADOR SINCRONO BINARIO

El contador síncrono binario tiene fundamentalmente como base de funcionamiento los biestables síncronos J-K, pudiéndose emplear cualquier otro tipo de biestable síncrono. Al tratarse de biestables síncronos, solo transmitirán el efecto de las entradas a las salidas cuando aparezcan los flancos de la señal de reloj. En la figura 10.1 se representa un esquema genérico para un contador de 4 bits.

Más específicamente para este caso en que se emplean biestables J-K, se activan empleando tanto los flancos de subida como los de bajada, al ser una estructura maestro-esclavo, tomando datos en el flanco de subida y transmitiéndolos a la salida en el flanco de bajada. La salida de cada biestable aparecerá por tanto coincidiendo con el flanco de bajada de la señal de reloj.

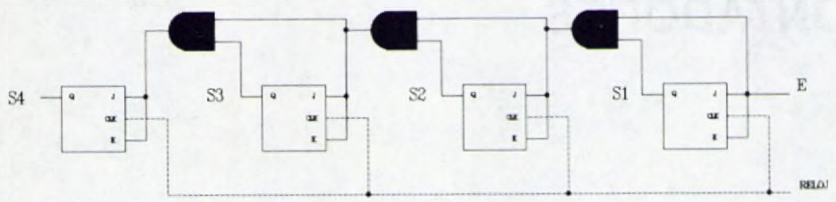


Figura 10.1. Contador síncrono de 4 bits

En el estado inicial todas las salidas están a cero y aunque las entradas de datos externos están activadas, no se transmiten a los biestables hasta que no aparezcan los flancos de la señal de reloj. Con el primer pulso de la señal de reloj, el primer biestable cambia su salida, obteniéndose "0001". Con los sucesivos pulsos de reloj, los biestables van cambiando a "10" y "1111", y así sucesivamente en forma cíclica, volviendo a comenzar el ciclo al cambiar de "1111" a "0000". La conversión de binario a decimal expresa más claramente la cuenta cíclica, es decir, tomará los valores 0, 1, 2, 3,..., 14, 15, 0, etc.

Existen tres opciones de contadores según el orden de cuenta que tengan establecido, siempre realizando funciones cíclicas de trabajo. Son los contadores CRECIENTES o ASCENDENTES (UP-COUNTERS) cuya salida es la suma creciente de los pulsos de entrada, los DECRECIENTES o DESCENDENTES (DOWN-COUNTERS) cuya salida es la suma decreciente de los pulsos de entrada, y los contadores CRECIENTE/DECRECIENTE (UP/DOWN-COUNTERS) cuya salida puede actuar en ambos sentidos.

Existen modelos de contadores UP/DOWN con una entrada de control, que para valor "0" realizan cuentas crecientes y para valor "1" cuentas decrecientes. Sin embargo el modelo más extendido es el dos entradas, una para cada opción de cuenta.

Cuando un contador debe ampliar el número de bits y se emplean circuitos integrados comerciales, se recurre a la conexión en cascada a través del terminal de acarreo (Ripple carry out) como los esquemas representados en la figura 10.2.

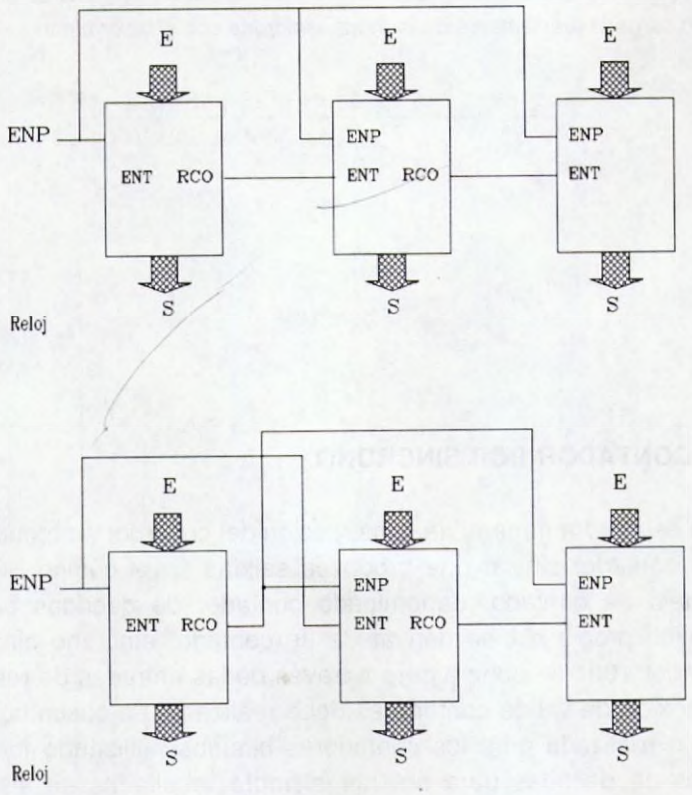


Figura 10.2. Contadores conectados en cascada

*## **Ejemplo 10.1**: Aplicar el cronograma de la figura 10.3 al contador síncrono de 4 bits y obtener gráficamente su salida.*

Solución:

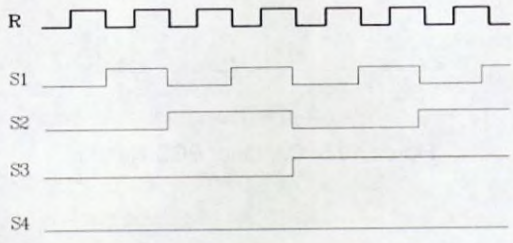


Figura 10.3

*## **Ejemplo 10.2**: Aplicar el cronograma de la figura 10.4 a un contador síncrono de 3 bits que ha sido cargado previamente de manera asíncrona con el valor binario "101".*

Solución:

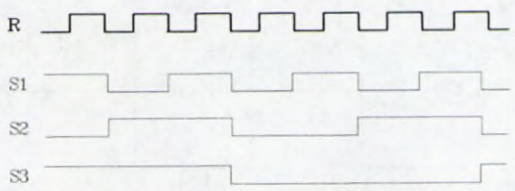


Figura 10.4

10.3. EL CONTADOR BCD SINCRONO

Con la estructura general de construcción del contador síncrono, se puede definir un contador similar que produzca salidas en el código BCD natural. Este modelo de contador denominado contador de décadas puede tener configuración propia o bien derivar de un contador síncrono binario que al llegar al valor 1001 se pone a cero a través de las entradas de reset.

La conexión de varios contadores debe realizarse en cascada de manera similar a lo realizado para los contadores binarios, utilizando forzosamente contadores de décadas para obtener siempre resultados en sistema BCD natural. En la figura 10.5 se representa un modelo genérico de contador BCD natural obtenido a través de la tabla de verdad del sistema BCD natural y simplificadas las ecuaciones mediante Karnaugh utilizando funciones incompletamente especificadas.

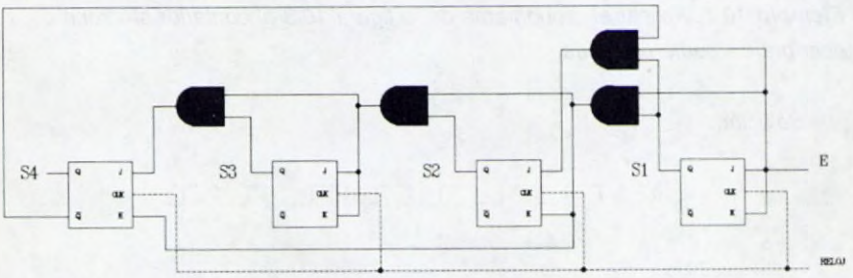


Figura 10.5. Contador BCD Natural

*## **Ejemplo 10.3**: Aplicar el cronograma de la figura 10.6 al contador BCD síncrono de 4 bits y obtener gráficamente su salida.*

Solución:

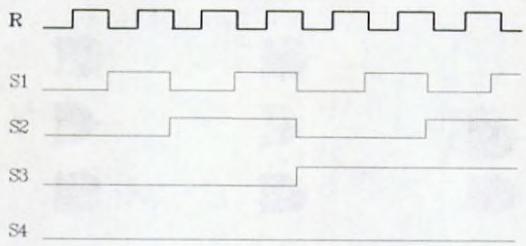


Figura 10.6

***## Ejemplo 10.4**: Aplicar el cronograma de la figura 10.7 a un contador BCD síncrono de 4 bits que ha sido cargado previamente de manera asíncrona con el valor binario "0110".*

Solución:

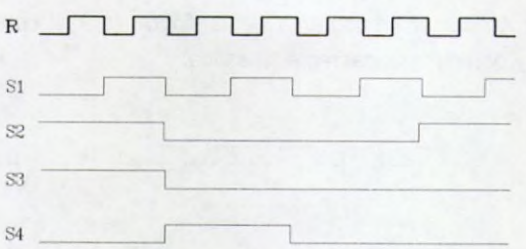


Figura 10.7

10.4. EL CONTADOR BCD EXCESO-3 SINCRONO

La estructura general de construcción del contador síncrono ya indicada en el contador BCD natural es la misma que se emplea para definir el contador BCD Exceso-3, igualmente diseñado con una configuración particular o bien derivando de un contador síncrono binario que al llegar al valor 1100 se pone a cero a través de las entradas de reset. La puesta a cero de este contador se realiza mediante la carga del valor 0011, correspondiente al cero en sistema BCD Exceso-3.

No es un contador excesivamente utilizado, siendo más útil a veces contar en binario o mediante décadas, convirtiendo los datos de partida y el resultado final al sistema particular que se desee emplear. Los comentarios realizados sobre la estructura y los montajes múltiples del contador BCD natural le son también de aplicación a este contador. En la figura 10.8 se representa un modelo genérico de contador BCD Exceso-3 obtenido a través de la tabla de verdad igual que el contador BCD.

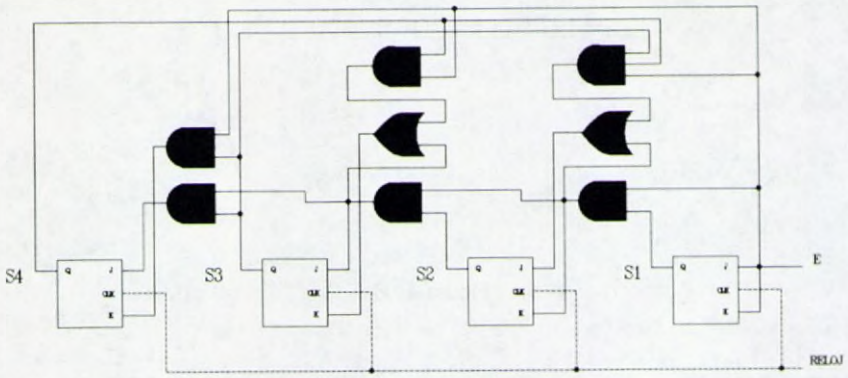


Figura 10.8. Contador BCD Exceso-3

***## Ejemplo 10.5**: Aplicar el cronograma de la figura 10.9 al contador BCD Exceso-3 síncrono de 4 bits y obtener gráficamente su salida.*

Solución:

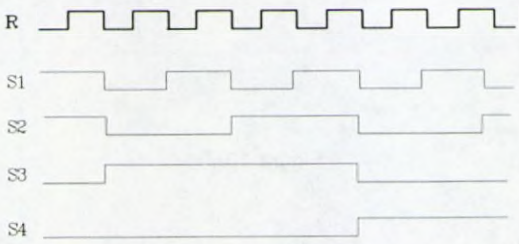


Figura 10.9

***## Ejemplo 10.6**: Aplicar el cronograma de la figura 10.10 a un contador BCD Exceso-3 síncrono que ha sido cargado previamente de manera asíncrona con el valor binario "0110".*

Solución:

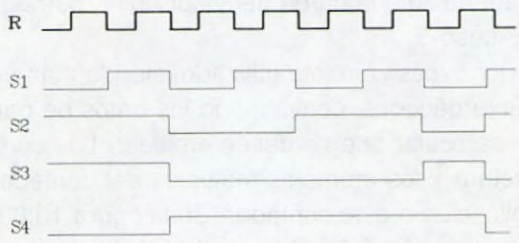


Figura 10.10

10.5. EL CONTADOR DE ANILLO

El contador de anillo es un circuito síncrono que realiza una cuenta cíclica. Sin embargo se diferencia de los anteriores contadores síncronos en que el resultado de la cuenta no se expresa numéricamente en forma directa, sino que la posición del desplazamiento cíclico es la que indica el valor de la cuenta.

El contador de la figura 10.11, representado por biestables D síncronos, transmiten la entrada a la salida cuando aparecen los flancos de activación de la señal de reloj. El biestable síncrono D transmite la entrada a la salida durante los flancos de la señal de reloj, excepto en los casos en que coinciden los flancos de las señales de entrada y de reloj, en los que transmite el valor de la entrada un instante antes de que aparezca el flanco de la señal de reloj.

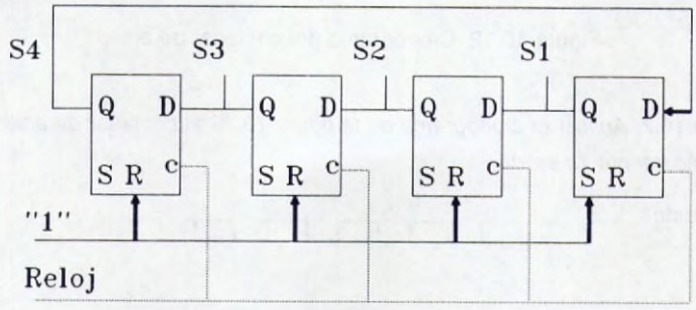


Figura 10.11. Contador de anillo

El contador es cíclico, realimentándose la última salida no complementada a la entrada del primer biestable. Para comenzar la cuenta, se necesita cargar el primer biestable, por lo que a través de la entrada de carga SET del primer biestable se introduce un "1". Al mismo tiempo, los otros biestables se ponen a cero a través de la entrada RESET. A medida que van apareciendo pulsos de reloj, las salidas S_1 a S_4 se van activando cíclicamente, indicando la cuenta correspondiente. El movimiento de los bits a través de las cuatro salidas S_4, S_3, S_2 y S_1 será el descrito en la siguiente tabla:

S_4	S_3	S_2	S_1
0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0

Tabla 10.1. Contador de anillo

Si se conectan diodos leds o displays siete segmentos a las salidas, se puede comprobar visualmente el resultado de la cuenta. El cronograma de funcionamiento del circuito se representa en la figura 10.12.

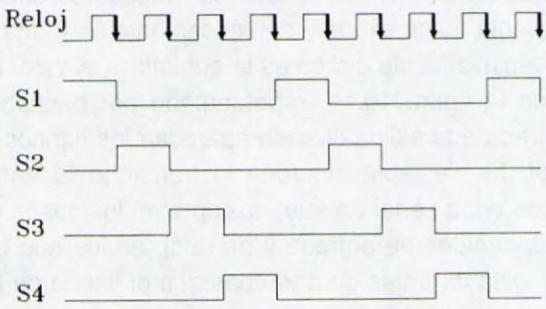


Figura 10.12. Cronograma del contador de anillo

*## **Ejemplo 10.7**: Aplicar el cronograma de la figura 10.13 al contador de anillo de 4 bits y obtener gráficamente su salida.*

Solución:

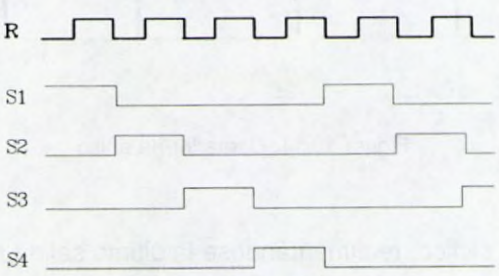


Figura 10.13

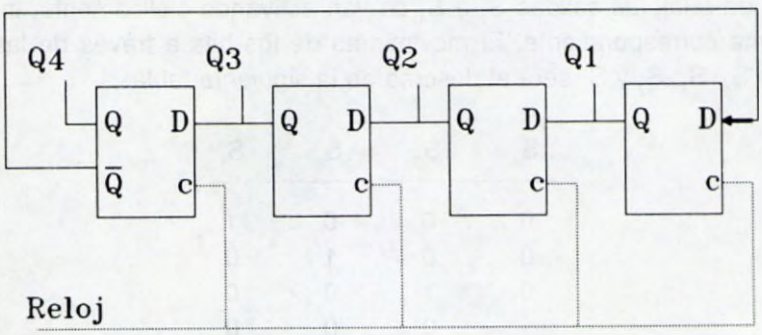


Figura 10.15. Contador conmutado

*## **Ejemplo 10.8**: Aplicar el cronograma de la figura 10.14 a un contador de anillo de 5 bits que ha sido cargado previamente de manera asíncrona con el valor binario "00100".*

Solución:

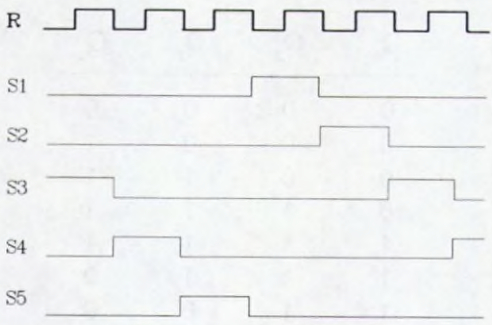


Figura 10.14

10.6. EL CONTADOR CONMUTADO EN COLA DE JOHNSON

El contador conmutado tiene un funcionamiento similar al analizado para el contador de anillo en el apartado anterior. Está constituido por un conjunto de biestables, conectados en cascada, con realimentación final de la última salida a la primera entrada. Sin embargo, la realimentación en este contador no procede de la salida Q del último biestable, sino de su salida complementada.

Eso significa que en el comienzo de la cuenta, si los biestables no han sido cargados previamente, en la última salida no complementada existirá un valor "0", y en la complementada un "1". Como este valor "1" es el que aparece en la entrada del primer biestable debido a la realimentación, no es necesario cargar previamente con un "1" el primer biestable como se hizo en el contador de anillo, puesto que al comenzar la señal de reloj a funcionar, el circuito realizará la cuenta normal.

En la figura 10.15 se representa un esquema de este contador con biestables D síncronos. Aunque no se incluye en el esquema para evitar confusión, se puede habilitar una entrada de puesta a cero de todos los biestables a través de su entrada de RESET. La tabla de salidas se representa a continuación, teniendo en cuenta que el estado inicial es el "0000" y que el contador funciona en forma cíclica.

El cronograma que representa el funcionamiento del contador conmutado se representa en la figura 10.16. Al existir ocho posibles salidas a través de 4 terminales S_4, S_3, S_2 y S_1, se conectan dichos terminales a la entrada de un decodificador, activándose 8 salidas, según se indica en el esquema de la

figura 10.17. Como este decodificador no es estandar, dadas las especiales combinaciones binarias de las entradas, se necesita diseñar una aplicación específica para este caso.

Q_4	Q_3	Q_2	Q_1
0	0	0	0
0	0	0	1
0	0	1	1
0	1	1	1
1	1	1	1
1	1	1	0
1	1	0	0
1	0	0	0

Tabla 10.2. Contador conmutado

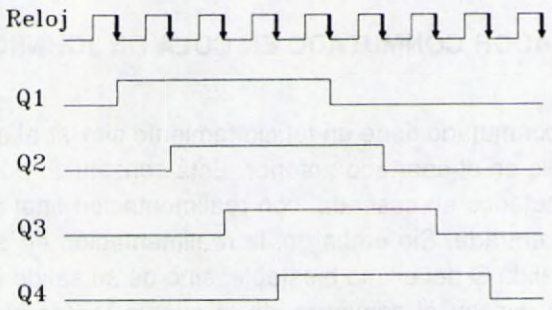


Figura 10.16. Cronograma del contador conmutado

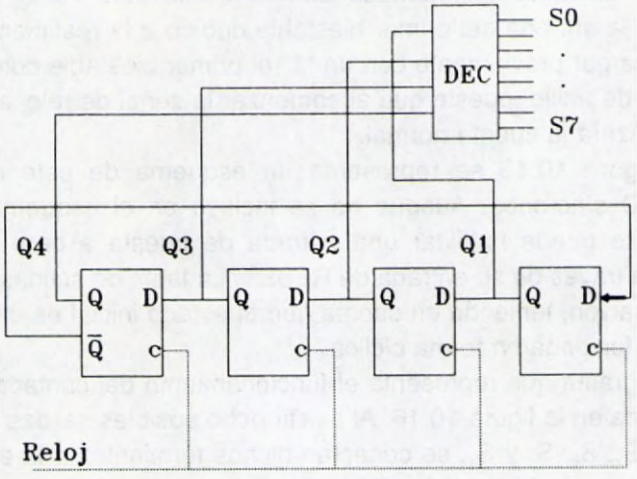


Figura 10.17. Contador conmutado completo

Para diseñar el decodificador se construye una tabla de verdad completa, donde se incluyan las entradas que produzcan situaciones inespecificadas. Para evitar problemas de nomenclaturas, se definirán como Q_4, Q_3, Q_2 y Q_1 los terminales de salida del circuito de biestables, y como S_7, S_6,..., S_0 las salidas del decodificador.

Q_4	Q_3	Q_2	Q_1	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	0	-	-	-	-	-	-	-	-
0	0	1	1	0	1	0	0	0	0	0	0
0	1	0	0	-	-	-	-	-	-	-	-
0	1	0	1	-	-	-	-	-	-	-	-
0	1	1	0	-	-	-	-	-	-	-	-
0	1	1	1	0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	-	-	-	-	-	-	-	-
1	0	1	0	-	-	-	-	-	-	-	-
1	0	1	1	-	-	-	-	-	-	-	-
1	1	0	0	0	0	0	0	0	1	0	0
1	1	0	1	-	-	-	-	-	-	-	-
1	1	1	0	0	0	0	0	1	0	0	0
1	1	1	1	0	0	0	1	0	0	0	0

Tabla 10.3. Decodificador de Johnson

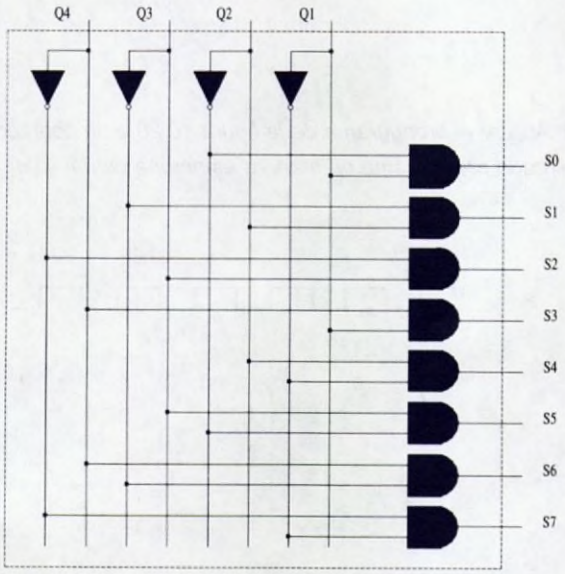


Figura 10.18. Decodificador del contador conmutado

Aplicando Karnaugh para funciones incompletamente especificadas, se obtienen las ecuaciones lógicas que determinan el circuito combinacional representado en la figura 10.18.

*## **Ejemplo 10.9**: Aplicar el cronograma de la figura 10.19 al contador conmutado de 4 bits y obtener gráficamente su salida.*

Solución:

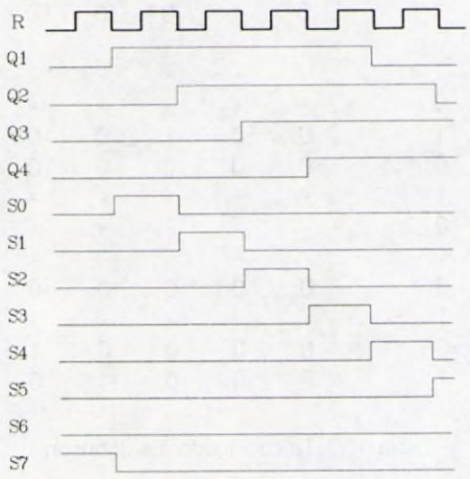


Figura 10.19

*## **Ejemplo 10.10**: Aplicar el cronograma de la figura 10.20 a un contador conmutado de 4 bits que ha sido cargado previamente de manera asíncrona con el valor binario "0010".*

Solución:

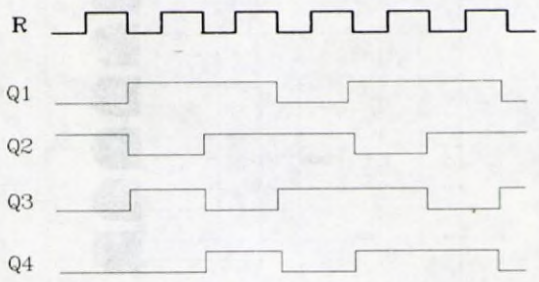


Figura 10.20

10.7. EL CONTADOR ASINCRONO

El contador asíncrono genérico se puede construir de idéntica forma que los contadores síncronos, partiendo de la tabla de estados asíncronos, simplificando y desarrollando por métodos secuenciales. Sin embargo, los circuitos que se emplean normalmente como contadores asíncronos, provienen de una modificación de los contadores síncronos y no como una constitución del circuito secuencial asíncrono correspondiente.

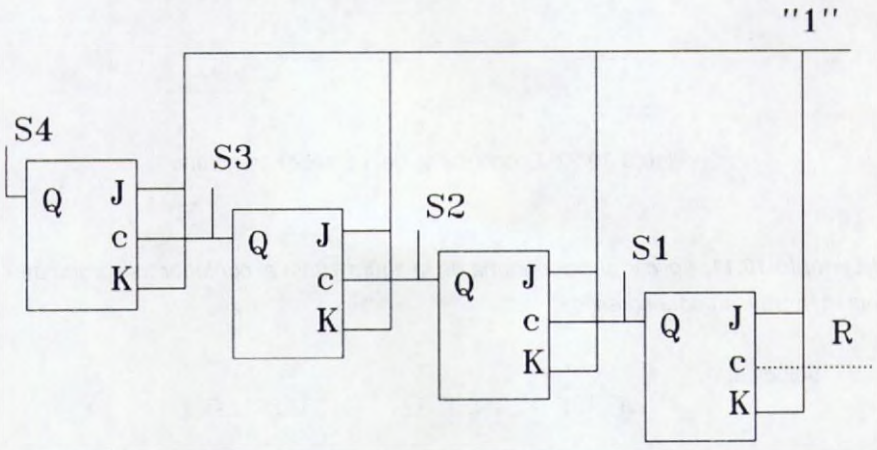


Figura 10.21. Contador asíncrono

Como ejemplo más representativo de dichos contadores, se analiza a continuación el circuito formado por biestables J-K, funcionando todos excepto el primero en forma asíncrona, es decir, no siendo activados por la señal de reloj, sino por las señales provenientes de las etapas previas. En la figura 10.21 se representa el esquema del contador asíncrono de 4 salidas.

En el primer biestable se introduce la señal de reloj, que equivale a la señal de entrada o de referencia cuyos pulsos han de ser contados. Sucesivamente, las salidas no complementadas de cada biestable constituyen las entradas de la señal de reloj del siguiente biestable. Al mismo tiempo, las entradas J y K de todos los biestables están conectadas a un valor "1" para que todos puedan funcionar en conmutación, dado que con entradas "1" simultáneamente en los terminales J y K el biestable cambia la salida de "0" a "1" ó de "1" a "0" cuando aparecen los flancos de activación en la señal de reloj.

Se utiliza una señal periódica (señal de reloj), como entrada en el primer biestable, pero se puede utilizar cualquier otra señal aperiódica. Las salidas de los biestables establecen una cuenta binaria ascendente, representada

mediante el cronograma de la figura 10.22. Mediante las entradas SET de cada uno de los biestables se pueden cargar inicialmente con un valor distinto de "0", y a través de las entradas RESET se puede realizar la puesta a cero del contador completo.

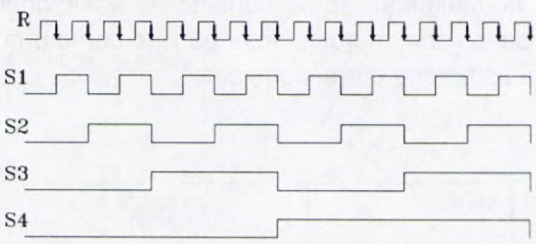


Figura 10.22. Cronograma del contador asíncrono

*## **Ejemplo 10.11**: Aplicar el cronograma de la figura 10.23 al contador asíncrono de 4 bits y obtener gráficamente su salida.*

Solución:

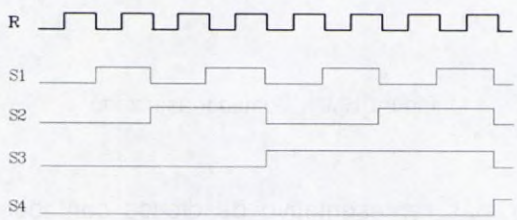


Figura 10.23

*## **Ejemplo 10.12**: Aplicar el cronograma de la figura 10.24 a un contador asíncrono de 4 bits que ha sido cargado previamente de manera asíncrona con el valor binario "0010".*

Solución:

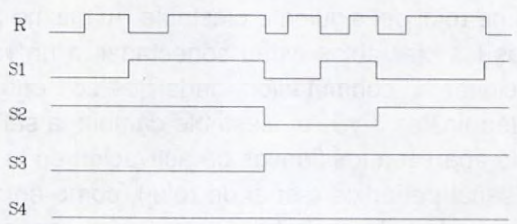


Figura 10.24

TEMA 11
REGISTROS

11

REGISTROS

11.1. INTRODUCCION

A lo largo de los diversos temas que componen el estudio de la lógica binaria y los circuitos digitales, tiene lugar la aparición de un término esencial en el tratamiento de la información que es el almacenamiento. Almacenar la información significa poder disponer del resultado de cualquier operación que se realice, poder utilizar ese resultado para etapas posteriores, etc.

Sin embargo, existe una etapa intermedia entre la creación o introducción externa de una información y el almacenamiento en las memorias. Esta etapa intermedia debe ser cubierta por elementos almacenadores de información, polivalentes según los distintos usos a los que el sistema puede destinarlos, y que se denominan REGISTROS. Los registros serán los elementos de la estructura interna de un computador que servirán de soporte temporal de la información.

En el presente tema se analizan los registros, haciendo especial mención a los registros de desplazamiento, a sus posibilidades de carga, puesta a cero, trasvase de información, comunicación con otros elementos, etc. Este análisis puede constituir una etapa previa en el análisis de estructuras más complejas como la de los sistemas basados en microprocesadores.

11.2. CONSTITUCION Y ESTRUCTURA INTERNA

Los registros son elementos de almacenamiento temporal de datos en sistema binario. Se han definido como los elementos de pequeña capacidad de almacenaje en la estructura de un sistema digital. No son sin embargo los elementos más pequeños de almacenaje, ya que esta descripción se podría asignar a los biestables, elementos que pueden almacenar un bit de información.

El límite más pequeño de un registro sería por tanto el biestable elemental, aunque normalmente se utilizarán registros formados por varios biestables. Por la estructura interna de los ordenadores y sistemas digitales, los registros de 8 y 16 bits son los más utilizados.

Los registros pueden dividirse en serie o paralelo, según la forma de procesar los bits de información. En la figura 11.1 se representan simbólicamente ambas posibilidades, para registros de 8 bits.

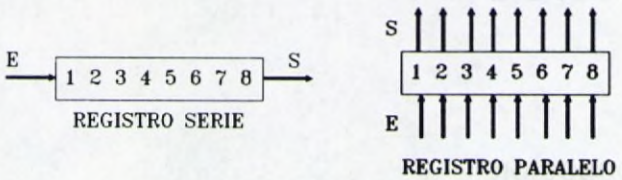


Figura 11.1. Registros serie y paralelo

El registro serie procesa los datos bit a bit, siendo éstos introducidos por un extremo y extraídos por el opuesto. El registro paralelo procesa simultáneamente todos los bits de un dato, tanto de entrada como de salida. (Al utilizar el término procesado de información se hace referencia tanto a grabación como a lectura de datos). La utilización de un tipo u otro de registro, depende de la operación a realizar, y también en determinada medida, de la relación tiempo/coste del sistema digital que se quiere diseñar.

La mayor velocidad de respuesta del registro paralelo sobre el serie va acompañada de un mayor coste de los elementos que procesan la información. La combinación de las dos posibilidades de funcionamiento de un registro, en serie y en paralelo, determina el llamado registro universal. Este registro está formado por biestables universales (J-K y S-R) cuyas conexiones, entradas, salidas y funcionamiento se indican en la figura 11.2.

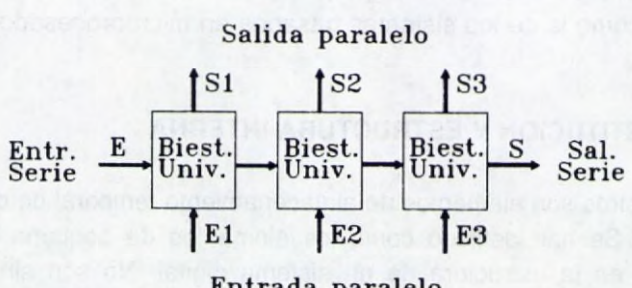


Figura 11.2. Registro universal

A continuación se indican brevemente algunas operaciones realizadas con los registros, como son la puesta a cero, carga, complementación y desplazamiento. En la puesta a cero, utilizando las entradas correspondientes S-R, se introduce un "1" en todas las entradas R (Reset) de los biestables simultáneamente, según se indica en la figura 11.3. La salida de todos ellos será "0" de acuerdo con sus características de funcionamiento.

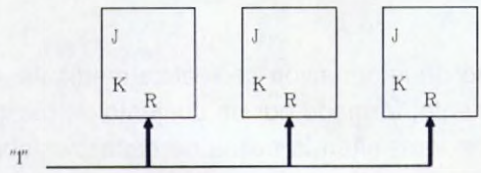


Figura 11.3. Puesta a cero

La carga se realizará dejando que las señales exteriores puedan ser introducidas en los biestables a través de los terminales S (Set), por lo que se dispondrán una serie de puertas de comparación AND accionadas simultáneamente por una señal de carga o de activación tal como se aprecia en la figura 11.4.

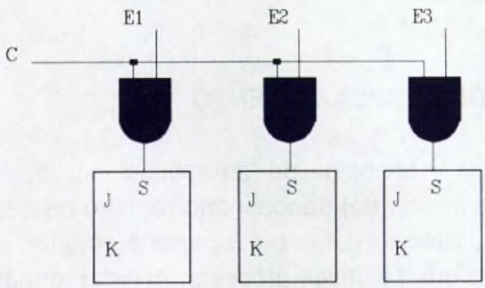


Figura 11.4. Carga previa

Al introducir un "1" en la señal de carga C, las salidas de las puertas lógicas AND serán iguales a las señales de las entradas E_1, E_2 y E_3. La función de complementación se realiza con las entradas correspondientes a los biestables J-K, conectando ambas entradas a una señal exterior de valor "1" tal como se aprecia en la figura 11.5.

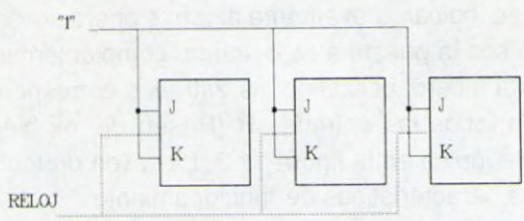


Figura 11.5. Complementación

El desplazamiento de información se realiza mediante el denominado registro de desplazamiento, formado por un conjunto de biestables, empleando las entradas J-K, que transmiten la salida de cada biestable al contiguo con cada pulso de la señal de reloj, según se observa en la figura 11.6. (Se omite la carga del primer biestable)

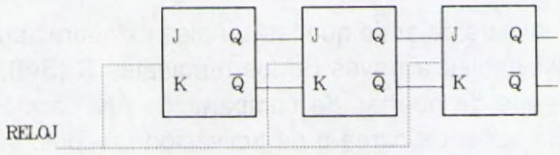


Figura 11.6. Desplazamiento

11.3. **REGISTROS DE DESPLAZAMIENTO**

La más importante y conocida de las aplicaciones de los registros es la que determina la existencia del denominado registro de desplazamiento. Basado en la propiedad determinada en el apartado anterior, su longitud depende del número de bits que se desee procesar. Al estar formados por biestables universales son circuitos síncronos y por tanto los cambios de información de un biestable a otro se producen sincronizados con la señal de reloj.

El desplazamiento puede realizarse a derecha o izquierda según la disposición de la entrada de datos. Así por ejemplo, el registro de 4 bits de la figura 11.7 desplaza los bits hacia la derecha perdiéndose la información del bit menos significativo en cada desplazamiento.

Al introducir un bit de información en el primer biestable B_1, la información almacenada en el instante anterior en B_1 pasará a B_2, la de B_2 a B_3, la de B_3 a B_4 y la de B_4 se perderá. A cada señal de reloj se producen todos

los desplazamientos simultáneamente. En la figura 11.8 se indica la evolución de los datos introducidos en un registro de desplazamiento puesto a cero inicialmente.

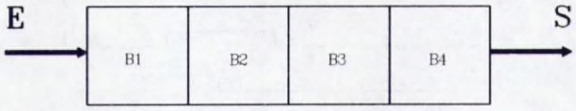


Figura 11.7. Registro de desplazamiento de 4 bits

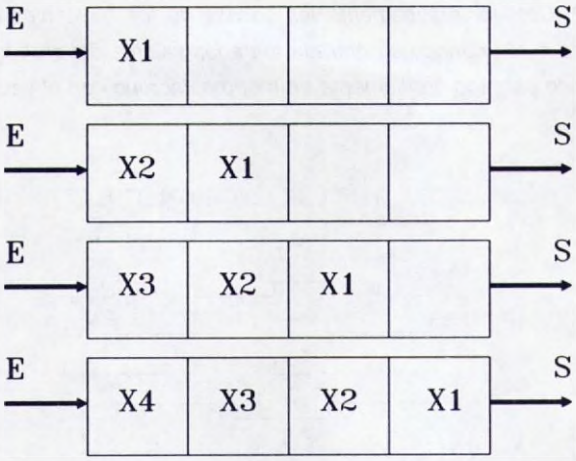


Figura 11.8. Etapas del desplazamiento

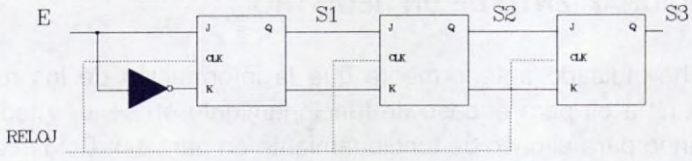


Figura 11.9. Constitución de un registro de 3 bits

*## **Ejemplo 11.1**: Obtener gráficamente las salidas de los biestables de un registro de desplazamiento de 4 bits cuando su entrada toma los valores del cronograma de la figura 11.10.*

Solución:

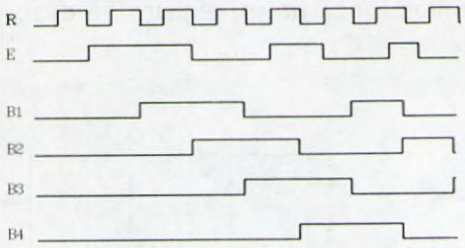


Figura 11.10

*## **Ejemplo 11.2**: Obtener gráficamente las salidas de los biestables de un registro de desplazamiento de 4 bits cuando su entrada toma los valores del cronograma de la figura 11.11 habiendo sido cargado previamente en modo asíncrono con el valor "0111".*

Solución:

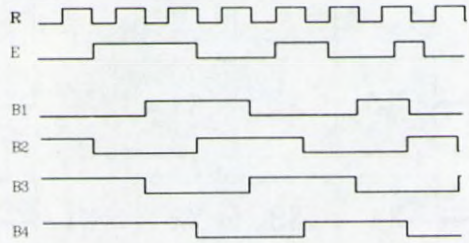


Figura 11.11

11.4. FUNCIONAMIENTO DE UN REGISTRO

Ya se ha indicado anteriormente que la información de los registros se procesaba bit a bit para el caso de funcionamiento en serie, y todos los bits en simultáneo para el caso de funcionamiento en paralelo. Esto ocasiona que en el caso de los registros de funcionamiento en serie, la información de cada biestable elemental tiene que ser modificada en cada desplazamiento de un bit al biestable contiguo.

Si se analiza con más detalle el funcionamiento en serie de un registro, se puede observar que la información inicialmente tiene dos posibles sentidos o movimientos, que se denominarán desplazamientos. Estos movimientos de información pueden ser hacia la derecha o hacia la izquierda, en función de

la entrada de datos que se seleccione. La información que entre en un registro, se pierde al extraerla del mismo. Al perderse la información, el registro se llena con ceros, lo que se denomina desplazamiento lógico. Si el registro se llena con el bit de signo, se denomina desplazamiento aritmético.

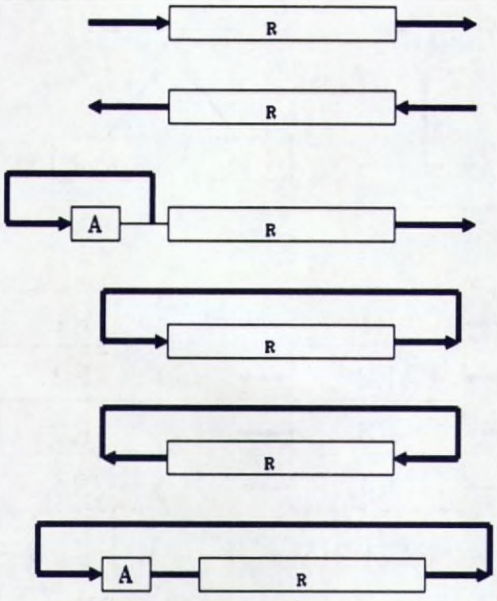


Figura 11.12. Desplazamiento de información

En caso de que no se quiera perder la información contenida en el registro, situación que ocurre en los desplazamientos indicados anteriormente, se puede efectuar la operación denominada rotación, rellenándose el registro con los mismos bits que van saliendo. Si en la rotación se incluye también el bit indicador del acarreo se denominará rotación con el acarreo. En la figura 11.12 se representan algunos esquemas simbólicos de funcionamiento de los desplazamientos que pueden tener lugar en un registro, teniendo en cuenta que puede haber sentido de movimiento a derecha o izquierda, con lo cuál, algunos casos indicados en dicha relación, pueden ser ampliados con el movimiento en sentido inverso.

Un registro puede conectarse con otros de varias maneras para transmitir información, según se indica en la figura 11.13. Si se agrupan varios registros, se pueden formar agrupaciones denominadas baterías o pilas, según que la información se transmita en paralelo a todos los registros simultánea-

mente o en serie. En la batería se seleccionan los registros mediante codificadores y decodificadores, y la información se transmite en serie. En las pilas, ya mencionadas anteriormente al tratar las memorias, se agrupan varios registros, transmitiéndose la información en modo LIFO (último en entrar, primero en salir).

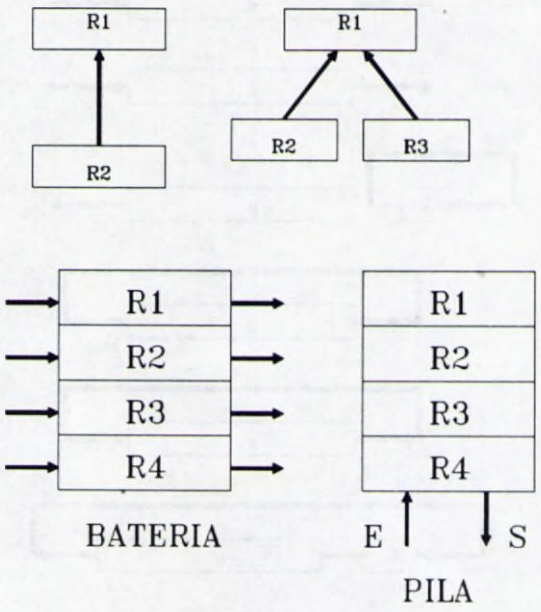


Figura 11.13. Conexión de registros

11.5. **FORMAS DE CONEXION**

Los registros pueden conectarse entre sí mediante estructura de calle o de estrella. La estructura de CALLE o BUS, representada en la figura 11.14, permite el intercambio de datos a través de una calle común a un determinado número de registros. El paso de los datos de un registro a la calle para proceder a la transmisión de información a otra parte del sistema, será controlado por la unidad central. El movimiento de información en cada registro es bidireccional.

Tiene el inconveniente de que no puede simultanearse el movimiento de información en la calle, y por lo tanto sólo puede enviarse información desde un registro en un instante de tiempo. La estructura en estrella o poligonal

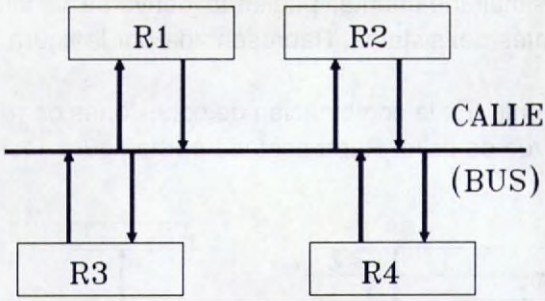


Figura 11.14. Estructura de bus

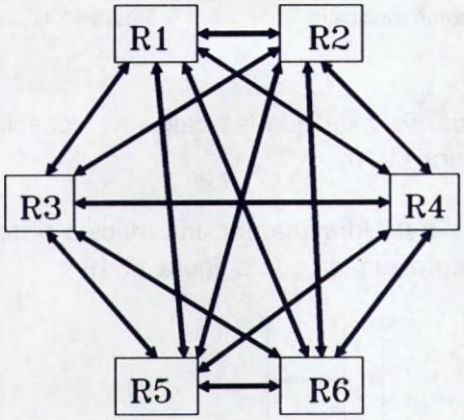


Figura 11.15. Estructura en estrella o poligonal

representada en la figura 11.15, conecta entre sí todos los registros de la estructura, pudiéndose realizar simultáneamente varias transferencias de datos entre registros.

Sin embargo este tipo de conexiones se complica enormemente en caso de que el número de registros a conectar entre sí sea bastante elevado. A partir de estas estructuras básicas, se pueden obtener otras posibilidades de conexión, sobre todo cuando se introduce la conexión sencilla o directa entre registros.

Sin necesidad de analizar si las conexiones se realizan en serie o en paralelo, sino simplemente como elementos genéricos que se conectan entre sí, se definen los nuevos montajes, que determinan las siguientes posibilidades de conexión de registros:

- Conexión de calle múltiple o multicalle: Es la conexión de un registro a varias calles simultáneamente, pudiendo derivarse la información a elementos diferentes del sistema. Representada por la figura 11.16.

- Conexión conjunta: Es la combinación de conexiones de registros en forma directa y a través de calle. Representada por la figura 11.17.

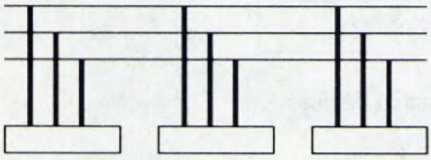


Figura 11.16. Conexión multicalle

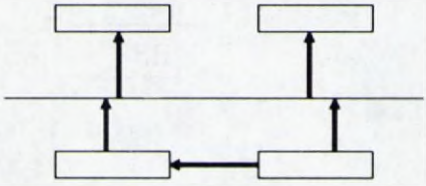


Figura 11.17. Conexión conjunta

- Conexión bifurcada: Permite que las calles se conecten entre sí. Representada por la figura 11.18.

- Conexión reticular: Está formada por una retícula o matriz de conexiones entre registros. Representada por la figura 11.19.

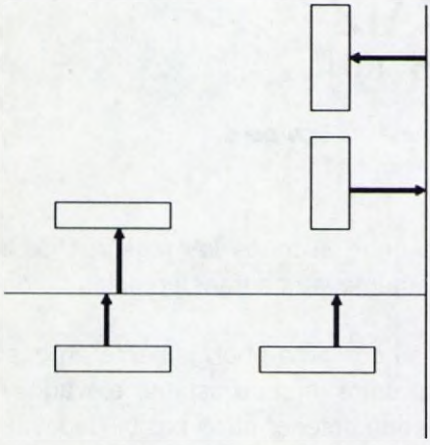


Figura 11.18. Conexión bifurcada

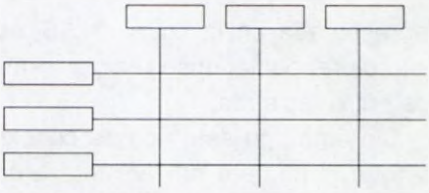


Figura 11.19. Conexión reticular o matricial

Las posibilidades de una u otra forma de conexión son diferentes, y el coste de las instalaciones también. Por ello, en un sistema digital es posible encontrar simultáneamente varias maneras distintas de conexionado, depen-

diendo del tipo de información que se procese y de los registros y elementos del equipo que estén implicados, como por ejemplo la conexión múltiple de la figura 11.20.

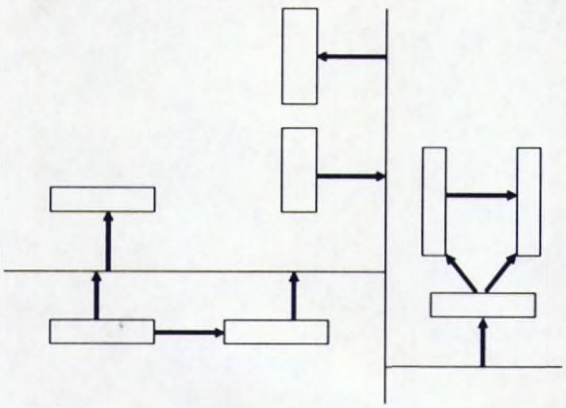


Figura 11.20. Conexión múltiple

11.6. UTILIZACION DE REGISTROS EN UNA UNIDAD CENTRAL

Como se ha indicado en apartados anteriores, los registros son definidos por un número de bits, que se denominará longitud de palabra. Este término es muy importante en la estructura y funcionamiento de un equipo informático, ya que determinará la constitución y tamaño de la memoria interna del mismo, la velocidad de procesamiento de la información, la constitución de los registros, etc.

De hecho, las unidades centrales de los computadores, base de los sistemas informáticos, se definen y conocen según la longitud de palabra que pueden procesar. Así por ejemplo, se pueden encontrar unidades centrales que procesan 8, 16, 32 bits. Si la unidad central procesa 8 bits, los registros que sirven de almacenamiento temporal a la información, serán principalmente de 8 ó de 16 bits, según que se pretenda almacenar una información en uno o dos campos para su posterior uso.

En función de esta longitud de palabra, y de su disposición y peculiar uso en la estructura interna del sistema informático, se asignan denominaciones particulares a algunos registros. Este es el caso de registros tales como el acumulador de resultados, el registro de instrucciones, el de direcciones, o el registro índice (de menor número de bits), almacenador de indicadores tales como el acarreo, el signo, rebosamiento, etc. Estos registros se definen con más detalle al tratar la arquitectura interna de los ordenadores.

TEMA 12
MEMORIAS

12

MEMORIAS

12.1. **INTRODUCCION**

Se describen a continuación unos elementos de almacenamiento de información, a los que en general podemos asignarles el calificativo de MEMORIAS. Sin embargo, el uso cada vez más generalizado de la informática y los ordenadores hace que estos elementos reciban denominaciones particulares, que al popularizarse inducen a una separación del grupo originario de elementos que motivó su diseño y creación.

El presente tema trata de dar una visión global de estos elementos y para ello, se realizará una división en dos bloques genéricos a los que denominaremos memorias internas de los sistemas de proceso y memorias de masas. Se considerarán memorias internas de los sistemas de proceso a los elementos que constituyen o forman parte de la arquitectura o equipo central de un sistema de proceso, computador, ordenador y cualquier otro sistema digital que requiera almacenamiento de información. (Se entiende por arquitectura de un sistema al conjunto de elementos que constituye la estructura básica de funcionamiento del mismo)

Dentro de este apartado se verán con más detalle las memorias de núcleos y las de semiconductores, que son las más empleadas en los sistemas informáticos. Se considerarán memorias de masas aquellos elementos de almacenamiento de un gran volumen de información. No forman parte del equipo básico o central del sistema aunque sí están conectados a él, realizando operaciones de transmisión bidireccional de información y se les denomina en general elementos periféricos. Se tocarán superficialmente en este tema, dando una breve descripción del uso, denominación y características. En resumen, se verán todas las características que definen las memorias de los sistemas centrales y al final se resumirán las memorias de masas.

12.2. CARACTERISTICAS DE LAS MEMORIAS

Una memoria por ejemplo de lectura y escritura (término que se detallará más adelante), tiene una serie de características de funcionamiento así como una serie de señales de entrada y salida que pueden evolucionar en función de determinados aspectos. Existe una entrada de dirección y otra para información a grabar, una salida de información y señales de control de lectura y escritura como mínimo. El esquema de la figura 12.1 es una de las maneras de simbolizar una memoria.

Asimismo, los parámetros básicos de la memoria dependen del tiempo, dado que la respuesta de un sistema depende de la velocidad de acceso a la información de la memoria. Se denomina tiempo de acceso el que se emplea desde la ejecución de la orden de activación (por ej. lectura) de la memoria hasta que se obtiene la información en la salida.

Capacidad de una memoria es el número máximo de palabras que puede almacenar y suele venir expresado en kilopalabras, siendo el valor de 1K=1024 palabras (potencia 10 de base 2). La longitud de palabra suele oscilar entre 8, 16, 32 y 64 bits en función del sistema informático, aunque también existen otros valores intermedios.

Una memoria es equivalente a un conjunto de registros donde se almacena información. Cada uno de los registros puede por tanto ser cargado o grabado con un determinado valor con lo que es fácil observar o leer lo que existe grabado en cada uno de ellos. Se necesita de esta forma un acceso para los datos a grabar, una salida de los datos a ser leídos y una forma de selección de la posición de memoria donde está grabada dicha información.

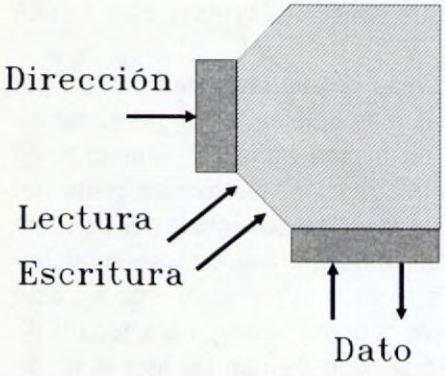


Figura 12.1. Símbolo de una memoria

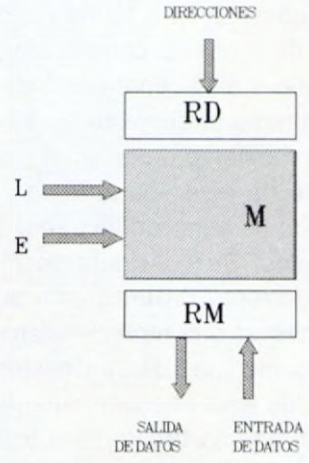


Figura 12.2. Estructura de trabajo de una memoria

Un esquema genérico de funcionamiento de una memoria puede ser el de la figura 12.2, donde la memoria tiene dos registros auxiliares de entrada y salida de información. El registro RD se denomina registro de direcciones de la memoria y el registro RM registro auxiliar de memoria. Al registro RD se envía la línea o posición de memoria (Dirección) donde se quiere leer o grabar un dato. Al registro RM se envía el dato que se quiere grabar en dicha posición de memoria o bien la memoria envía a RM el dato leído para que se pueda utilizar en otra zona del equipo digital. En cada caso el circuito que controla la memoria ejecutará un ciclo de lectura o escritura para realizar la operación correspondiente.

12.3. **CLASIFICACION DE LAS MEMORIAS**

Partiendo de los comentarios realizados en la introducción sobre los 2 bloques de memorias, existen diversos puntos de vista desde los que se pueden clasificar las memorias que constituyen el sistema central de un equipo informático, que a partir de ahora denominaremos Unidad Central. Este tipo de memorias son las que comúnmente se suelen denominar como tales, dándoseles a los elementos de almacenamiento masivo otra denominación.

Pueden dividirse según la forma de acceso, la funcionalidad, la dependencia de una fuente exterior de alimentación y las características temporales de mantenimiento de información.

Por la forma de acceso pueden clasificarse en:

- Serie o secuencial
- Paralelo o aleatoria

Por la funcionalidad pueden clasificarse en:

- De lectura y escritura
- Sólo de lectura

Por la dependencia de una fuente exterior pueden clasificarse en:

- Volátiles
- No volátiles

Por las características temporales de almacenamiento pueden clasificarse en:

- Estáticas
- Dinámicas

12.3.1. **Memorias de acceso serie**

Se pueden considerar como una agrupación de registros, que son leídos o grabados ordenada o secuencialmente. Esta secuencia de lectura o grabación de información se puede realizar de 2 formas según las necesidades del tratamiento de información, esquematizándose según se indica en la figura 12.3, y siendo sus denominaciones y sus características las de COLA y PILA o en términos ingleses FIFO y LIFO.

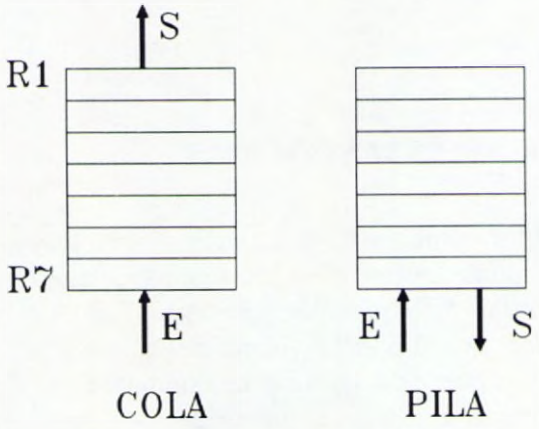


Figura 12.3. Estructuras LIFO y FIFO

COLA es una memoria en la que la lectura se realiza en el mismo orden que la grabación. Se denominan también FIFO (First-in- first-out), que quiere decir que la primera información que se graba es la primera que se lee, o lo que es lo mismo, la primera información que entra en memoria es la primera que sale.

PILA es la memoria en la que la lectura se realiza en orden inverso al que se realizó la grabación. Se denominan también LIFO (Last-in-first-out), que quiere decir que la última información que se graba es la primera que se lee, o lo que es lo mismo, la última información que entra en memoria es la primera que sale.

12.3.2. **Memorias de acceso aleatorio**

Su denominación RAM (Random access memory) y sobre todo el uso de este término, no es del todo correcto, dado que bajo la denominación de

memoria de acceso aleatorio no solo se encuentran las usuales RAM, sino también la familia ROM que se describirá más adelante.

El acceso aleatorio de información se realiza a través de unos indicadores de posición (direcciones). De esta forma un determinado dato se graba en una posición de memoria que estará referenciada con lo que denominamos dirección y que servirá posteriormente para su localización y lectura. Las posiciones de memoria no están condicionadas, sino que son totalmente aleatorias (al azar) y disponibles para que el programador las utilice según sus necesidades. En el grupo de memorias de acceso aleatorio se pueden citar las memorias RAM (o RWM) y las ROM.

12.3.3. **Memorias de lectura-escritura**

Se denominan RWM (Read-write-memory) y tienen la posibilidad de ser leídas y grabadas ("escritas" según la traducción literal del inglés) un número casi ilimitado de veces. Las memorias RAM son de lectura-escritura y constituyen las memorias centrales de los ordenadores y computadores, utilizándose como memorias temporales de almacenamiento de datos e instrucciones durante la ejecución de un programa.

12.3.4. **Memorias sólo de lectura**

Se denominan ROM (Read-only-memory) y sólo tienen la posibilidad de ser usadas como memorias de lectura un número casi ilimitado de veces después de haberse realizado el proceso de grabación. Se establecen 3 denominaciones genéricas correspondientes a 3 modelos distintos de memorias de sólo lectura:

- ROM: Se denomina así al grupo genérico y al modelo específico de memoria que ha sido grabada en el proceso de fabricación. Su constitución, normalmente de diodos y su conexionado electrónico interior, determinan las características y resultados de la misma.

- PROM: (Programmable-read-only-memory). Memoria programable sólo de lectura, es similar su constitución a la ROM grabada en el proceso de fabricación, pero permite que dicha grabación se realice en el instante deseado por el usuario. La técnica empleada es similar al paso de fuertes corrientes a través de fusibles, es decir, que se destruyen determinados

componentes electrónicos en función de la información a grabar en la memoria. Por ello, el ciclo de grabación es único y no permite modificaciones.

- RPROM: (Reprogrammable-read-only-memory). Es una memoria ROM reprogramable. Quiere decir que se puede borrar y volver a grabar. Podría por tanto asimilarse a una memoria de lectura y escritura, pero el tiempo y la energía necesarios para realizar estas operaciones es muy elevado comparado con las memorias normales RWM.

Están constituidas por transistores de tecnología MOS y el proceso de borrado suele realizarse mediante métodos eléctricos, denominándose EEROM, o bien por radiación ultravioleta, denominándose EPROM, siendo necesarios bastantes minutos de exposición en equipos externos al sistema informático.

12.3.5. **Memorias volátiles y no volátiles**

Son memorias volátiles las que pierden la información contenida cuando se interrumpe el suministro de energía. Las memorias RAM son el más claro exponente de este tipo de memorias. Son memorias no volátiles las que mantienen la información grabada a pesar de cesar el suministro de energía al sistema. En este apartado se encuentra el grupo de memorias ROM, cuyo contenido de información permanece inalterable a pesar de separarse del equipo al que está conectado.

Debido al problema de pérdida de información en el caso de las RAM es necesario tener en consideración la posibilidad de instalar equipos autónomos (baterías) de suministro de energía en caso de interrupción accidental del mismo. (Se denominan SAT o SAI-Sistemas de alimentación ininterrumpida y suministran energía durante varios minutos, tiempo suficiente para pasar la información de la RAM a otro soporte no volátil).

12.3.6. **Memorias estáticas y dinámicas**

Son memorias estáticas las que conservan la información sin que sufra degradación, deterioro o pérdida con el paso del tiempo. Se basan en circuitos con biestables que mantienen la señal de sus salidas dentro de los niveles lógicos de funcionamiento.

Son memorias dinámicas las que necesitan cada cierto intervalo de tiempo regenerar la información contenida en las mismas para evitar su pérdida.

Se basan en circuitos con pequeñas capacidades inducidas que necesitan regenerarse para evitar la descarga que con el tiempo tiene lugar en las mismas. Las memorias dinámicas tienen mayor posibilidad de almacenamiento de información que las estáticas, pero a cambio poseen otros inconvenientes y la mayor complejidad de sus circuitos de regeneración.

12.4. **MEMORIAS DE NUCLEO MAGNETICO**

Aunque ya no se emplean estas memorias, constituyen un ejemplo de fácil comprensión y una curiosidad "histórica" ya que se empleaban fundamentalmente en las unidades centrales de los primeros sistemas informáticos. Están constituidas por pequeños núcleos de ferrita, cuyo ciclo de histéresis tiene forma casi rectangular, lo que puede asemejarse a niveles "0" y "1" lógicos de información.

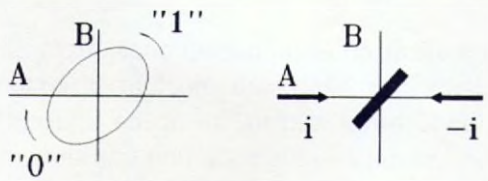


Figura 12.4. Memoria elemental de núcleo magnético

Estos pequeños núcleos tienen forma de anillo y están atravesados por cables a través de los que circula corriente, lo que equivale a la obtención de un efecto magnético similar al de un devanado de transformador. En la figura 12.4 se representa simbólicamente. La corriente de circulación provoca el efecto de magnetizar el núcleo de ferrita, de acuerdo con su ciclo de magnetización o histéresis. En la figura 12.5 se representa dicho ciclo.

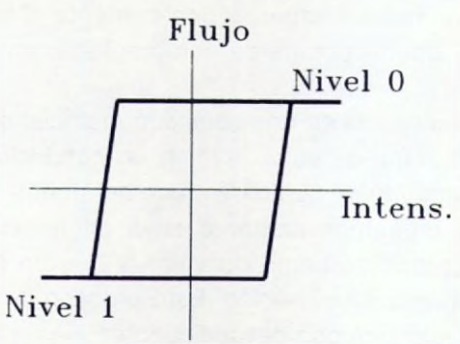


Figura 12.5. Ciclo de histéresis de una memoria

Este ciclo, en función del sentido de la corriente inducida en el núcleo, tiene 2 valores opuestos que son inalterables si no cambia la corriente de magnetización y que se hacen equivalentes al "1" y "0" lógico. Por tanto, un núcleo se queda "cargado" con un nivel lógico determinado al paso de la corriente eléctrica y no cambia de nivel lógico hasta que no cambie al valor opuesto la corriente eléctrica que realizó la carga. Por la propiedad de mantener la información se consideran no volátiles, siendo además estáticas al no necesitar regeneración.

Si dos cables conductores atraviesan un núcleo de ferrita tal como se indica en la figura 12.4, se pueden observar los siguientes aspectos:

- Por el cable "A" circula la corriente "i" que magnetiza el núcleo, determinando por ej. el nivel lógico "1". Este cable se denominará primario, siendo el segundo cable "B" el secundario (equivalencia con un transformador).

- El cambio de magnetización en el núcleo ocasionará una corriente inducida en el secundario. Por ello, para efectuar la lectura del nivel lógico (información) grabado en el núcleo, se aplica una corriente "-i" al cable conductor "A", con lo cuál se establece una corriente inducida en "B" (resultado de la lectura) puesto que estaba en el nivel "1" y pasa al "0".

- Si se necesita mantener grabada la información leída es necesario volver a aplicar una corriente "i" en el cable conductor "A".

Si en vez de usar un solo núcleo se realiza la conexión de un gran número de ellos para formar una memoria, las conexiones entre núcleos y conductores se realizan en forma de matriz, siendo atravesados los núcleos por 2 o más conductores según los modelos de memorias existentes, obteniéndose resultados diversos. Esta conexión en matrices obedece al hecho de que si un núcleo necesita una corriente "i" para cambiar de estado, esta corriente puede obtenerse como solapamiento de 2 corrientes de valor "i/2".

Esto quiere decir que si en una conexión matricial como en la figura 12.6, introduciendo una señal de valor "i/2" en un conductor horizontal y en uno vertical simultáneamente, el punto de cruce de ambos conductores identificará al núcleo que queremos cargar o cuya información se quiera leer. Al aplicar únicamente una corriente de valor "i/2" a un conductor horizontal o vertical no se magnetizará el núcleo. Este proceso de selección o identificación de líneas se realizará con decodificadores.

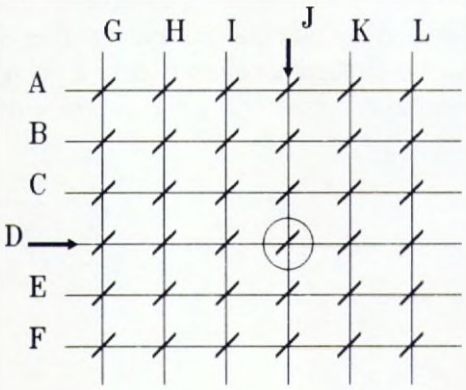


Figura 12.6. Matriz de núcleos

Según esta estructura matricial, pueden existir varias disposiciones de los núcleos y los conductores. La disposición 2D indica que cada núcleo de ferrita es atravesado por 2 cables conductores, tal como se ha descrito anteriormente. El esquema genérico de esta disposición es el representado en la figura 12.7, siendo la figura 12.8 la que representa la selección de un núcleo elemental.

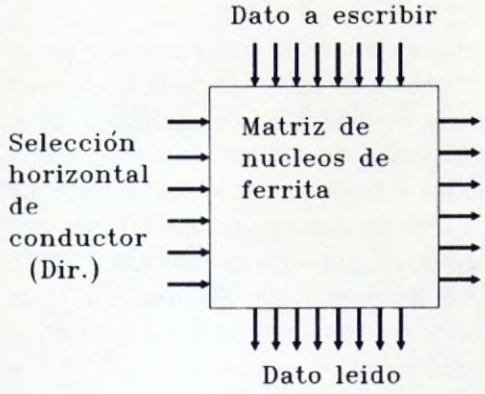


Figura 12.7. Disposición 2D

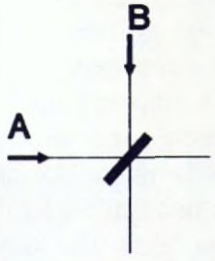


Figura 12.8. Selección de un núcleo 2D

La disposición 2 1/2 D indica que los núcleos de ferrita están atravesados por 3 cables conductores. Son 2 cables conductores de selección de núcleos y un tercer cable conductor denominado sensor, que atraviesa todos los núcleos de la matriz y realiza la lectura, de forma que atraviesa a la mitad de los

núcleos en un sentido y a la mitad restante en el sentido opuesto, con lo cuál, la posible corriente inducida sería nula. Su esquema es el de la figura 12.9 y la selección de un núcleo elemental se representa en la figura 12.10 siendo A y B los hilos selectores de corriente (i/2) y el hilo C el sensor o hilo de lectura.

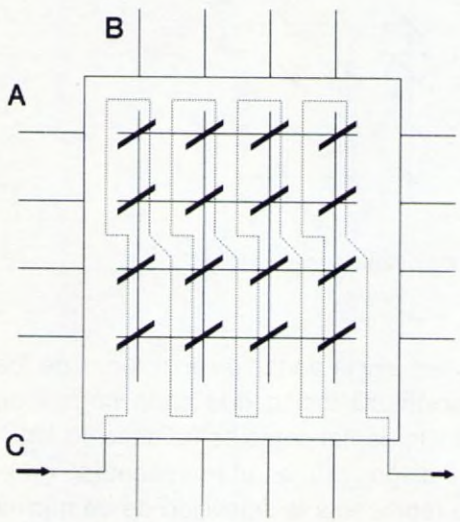


Figura 12.9. Disposición 2-1/2-D

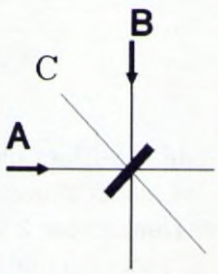


Figura 12.10. Selección de un núcleo 2-1/2-D

La disposición 3D indica que cada núcleo de ferrita es atravesado por 4 cables conductores. Se mantienen los 2 cables conductores de selección del nùcleo de ferrita, siendo el tercero un cable de inhibición y el cuarto un cable de lectura. El cable de inhibición atraviesa todos los núcleos de ferrita de una matriz y se emplea para anular una de las 2 posibles corrientes "i/2" de los cables conductores de selección. El cable de lectura atraviesa todos los núcleos de la matriz, por filas o columnas, igual que en la disposición 2 1/2 D, no siendo necesario anular la posible corriente inducida. Su esquema es el de la figura 12.11 y la selección de un núcleo elemental se representa en la figura 12.12 siendo A y B los cables selectores de corriente (i/2), C el hilo inhibidor y D el hilo de lectura.

12.5. **MEMORIAS DE SEMICONDUCTORES**

Igual que las memorias de núcleos de ferrita, las memorias de semiconductores se emplean fundamentalmente en las unidades centrales. Suelen ser de tipo estático y las más usuales están normalmente constituídas

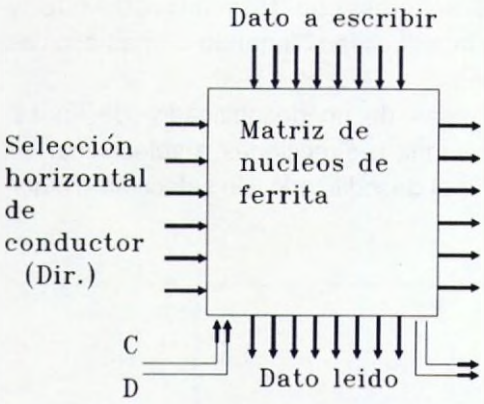


Figura 12.11. Disposición 3D

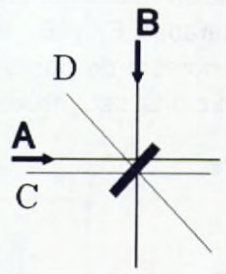


Figura 12.12. Selección de un núcleo 3D

por diodos, transistores y agrupación de estos elementos en forma de biestables. Dependiendo de que las memorias reciban o no suministro de energía independiente de la alimentación exterior, pueden ser clasificadas en volátiles o no volátiles.

La selección de los elementos se realiza igual que con los núcleos de ferrita a través de estructuras matriciales y de disposiciones equivalentes 2D y 3D. El esquema de la disposición 2D para redes de diodos es el representado en la figura 12.13.

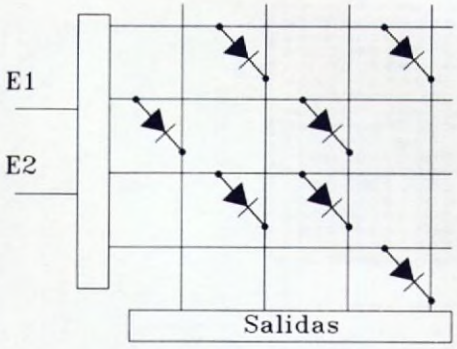


Figura 12.13. Estructura matricial 2D con diodos

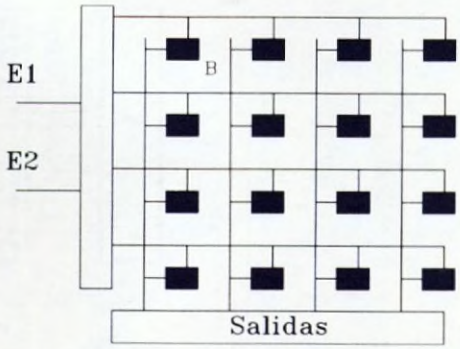


Figura 12.14. Estructura matricial 2D con biestables

Igualmente, el diagrama correspondiente a la disposición 2D para biestables, considerando a éstos (B) como bloques genéricos inespecificados conectados a una estructura matricial de conductores, sería el representado

en la figura 12.14. Estas configuraciones con transistores MOS y semiconductores son las utilizadas en la actualidad, llegando a unas escalas de integración realmente espectaculares.

Las entradas E_1 y E_2 son las entradas de un decodificador de líneas. Según el número de posiciones de memoria o direcciones a seleccionar en un circuito digital será necesario diseñar el decodificador de selección (Regis-

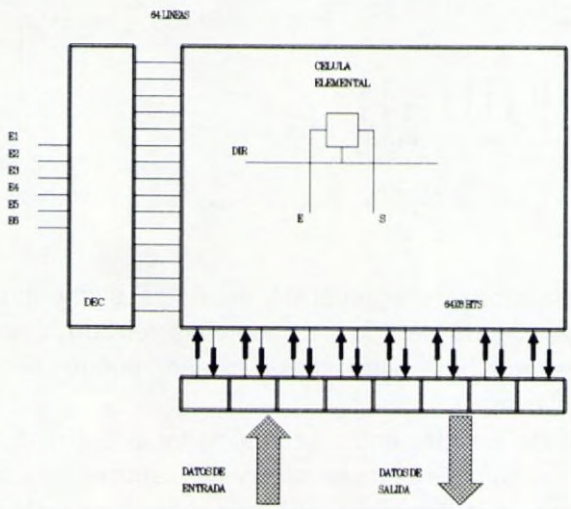


Figura 12.15. Memoria matricial de 64 bytes

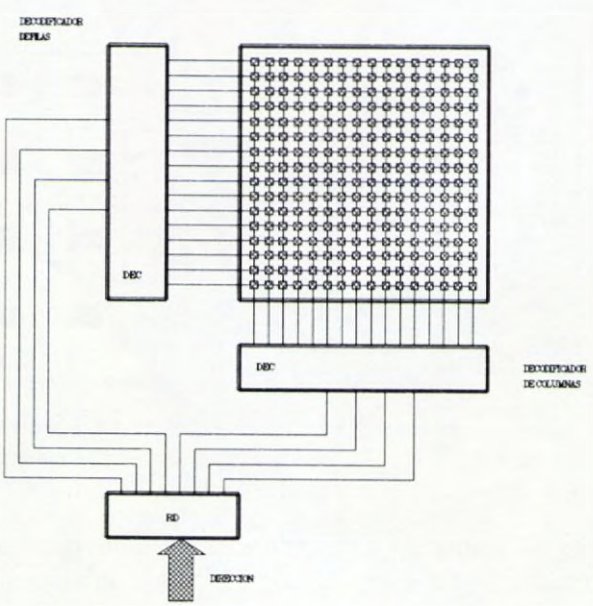


Figura 12.16. Decodificación de filas y columnas

tro de direcciones). Por ejemplo para una matriz de 64 líneas será necesario un decodificador de 6 entradas. Si cada línea tiene 8 biestables (8 bits), se tiene entonces una memoria de 64 bytes como en el diagrama de bloques de la figura 12.15.

Cada biestable elemental de este tipo de estructuras tiene el conexionado externo que se representa en la figura 12.17.a, utilizando biestables tipo D, donde un terminal L/E de lectura/escritura determina que para valor "0" se puede escribir y para valor "1" se puede leer la salida del biestable siempre que sea seleccionado mediante los decodificadores.

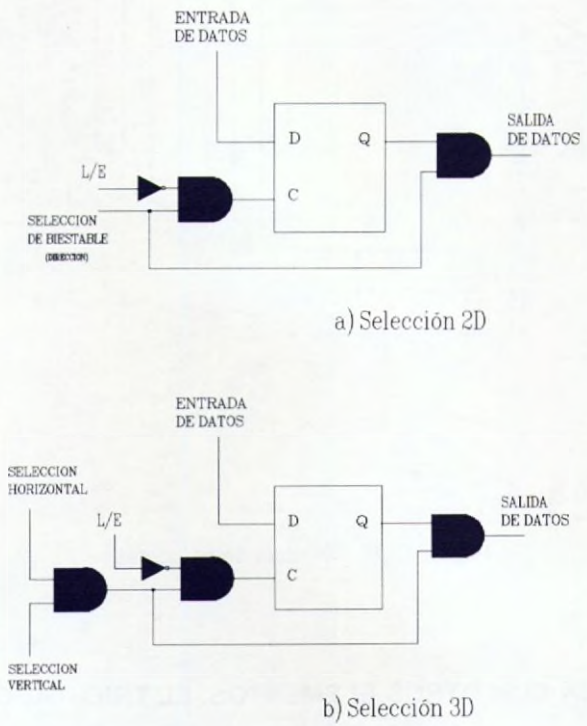


Figura 12.17. Biestable elemental de una memoria

Para aumentar la capacidad de las memorias se recurre a otras estructuras como la de disponer dos decodificadores para filas y columnas. En una estructura plana como en la figura 12.16 se selecciona cada uno de los bits elementales de la matriz, direccionándose estos en forma 3D como indica el esquema 12.17.b. Esta ampliación de la decodificación permite que en un montaje de varias estructuras matriciales paralelas se pueda obtener un bit de cada una de ellas. Formando por ejemplo un conjunto de 8 estructuras matriciales planas paralelas conectadas en paralelo a las salidas del

decodificador horizontal (de filas) se seleccionan simultáneamente 8 bits, formando por tanto una palabra o byte. En la figura 12.18 se representa simbólicamente una estructura de 8 bloques de 64 bytes, formando una memoria direccionable de 512 bytes ó 0,5K bytes. Los 9 bits necesarios para decodificar las filas y las columnas proceden del registro de direcciones RD simbolizado en la figura 12.2.

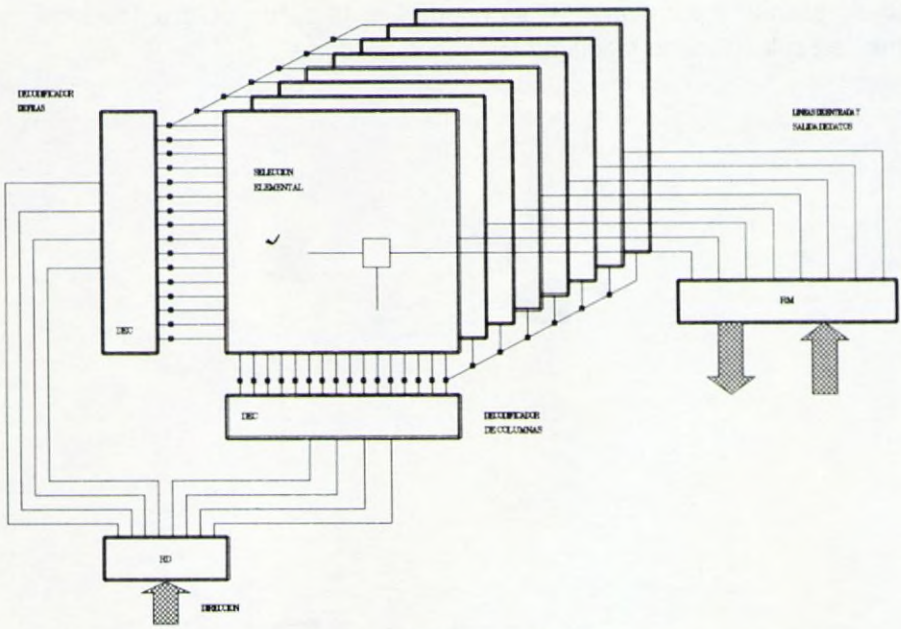


Figura 12.18. Memoria de 512 bytes

12.6. **CONEXION CON OTROS ELEMENTOS. EL TRIESTADO**

La información existente en la memoria y en los registros de cualquier equipo digital no se transmite normalmente en forma asíncrona, sino que existen una señales de control que realizan dichos trasvases. Para bloquear los posibles accesos a los distintos elementos de un sistema digital se emplean los elementos denominados TRIESTADOS (Three state buffer), cuyo símbolo se representa en la figura 12.19.

Cuando la entrada de control (C) o de habilitación del triestado es "0", la entrada se transmite normalmente a la salida, es decir, si E=0 S=0 y si E=1 S=1. Cuando la entrada de control es "1" el circuito se encuentra con un elemento en ALTA IMPEDANCIA. Esto significa que la resistencia que dicho

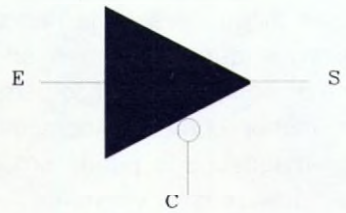


Figura 12.19. Símbolo del triestado

elemento ofrece al paso de la corriente es tan elevada que su efecto es como el de desconectarlo del resto del sistema, por lo que ni recibe ni envía señal alguna. Esta es la situación de espera hasta que se le permita conectarse con otros elementos. Todos los movimientos de información entre las memorias y los restantes componentes se realizan siempre a través de triestados.

12.7. **OTROS TIPOS DE MEMORIAS**

Aunque son menos utilizadas que las anteriores, existen otros tipos como las de hilos magnéticos planos, de burbujas magnéticas, CCD, etc., a medio camino entre el uso en unidades centrales y periféricos (de masa). La memoria de hilos planos está formada por barritas conductoras que hacen contacto transversalmente con otros conductores planos, como se representa en la figura 12.20. La selección se realiza igual que en los núcleos de ferrita, y tienen la ventaja de no perder la información al realizar el ciclo de lectura.

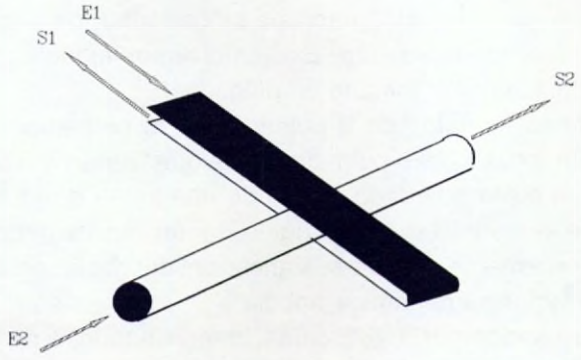


Figura 12.20. Memoria de hilos planos

La memoria de burbujas magnéticas es la denominación de los campos magnéticos de forma circular que se forman en una placa de material ferromagnético al aplicarle un campo magnético perpendicular. Tienen la ventaja del menor volumen, menor energía y comportamiento como memoria estática ya que el campo magnético lo puede producir un imán. Presentan una gran capacidad de almacenamiento no volátil.

Los dispositivos acoplados por carga CCD (Charge-Coupled-Device) son memorias de acceso serie y característica FIFO. Son unas agrupaciones de registros de desplazamientos con densidades de integración muy elevadas y tiempos de acceso muy cortos. Por su bajo coste y potencial futuro, es un buen candidato a sustituir a los elementos externos de almacenamiento de información (discos, cintas, etc.), aunque salvando adecuadamente el inconveniente de su volatilidad.

12.8. **MEMORIAS DE MASAS**

Son elementos de una gran capacidad de almacenamiento de información. Van acompañados de otros elementos accesorios para facilitar la transferencia de información, tales como unidades de intercambio, interfases (conjunto de señales y cable de conexión) y controladores de periféricos.

Suelen ser conocidos por ser elementos externos de los equipos informáticos y en la mayor parte de los casos, manipulables por el usuario. Suelen presentar la forma de discos, cintas, tarjetas y tambores magnéticos.

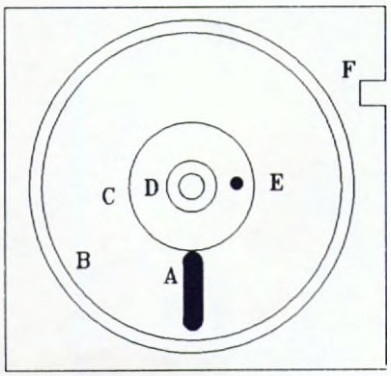
12.8.1. **Disco flexible**

Conocido también como floppy disk, es el dispositivo de almacenamiento más popular y mejor aceptado, gracias a su fiabilidad, bajo coste, rapidez de respuesta y su fácil manejabilidad. Existen diversos modelos según los equipos y se referencian por el tamaño en pulgadas.

Así se fabrican formatos de 8 pulgadas, 5,25 pulgadas (estándar para ordenadores personales), 3,5 pulgadas (nuevo estándar) y 3 pulgadas. Están formados por un disco guardado dentro de una funda protectora cuadrada o rectangular, de material flexible o rígido según los modelos y los usos y recubiertas interiormente de una sustancia antidesgaste, de bajo coeficiente de fricción (en inglés se denomina "media").

Pueden ser usados por 1 ó 2 caras, denominándose de simple o doble cara (single-sided o doubled-sided), con densidad de grabación simple o doble, lo que se expresa como simple densidad o doble densidad, pudiendo

incluso llegarse a una cuádruple densidad (simple-density, double-density, high-density). El formato de grabación sectorizado puede realizarse mediante hardware o mediante software (hard-sectored o soft-sectored). Un ejemplo de los mismos se representa en la figura 12.21, donde se ha esquematizado el más conocido y empleado de los discos flexibles, el de 5,25 pulgadas.



A= Contacto con el cabezal
B= Pista 0
C= Pista 39
D= Anillo de protección
E= Indice (comienzo de pista)
F= Protección contra borrado

Figura 12.21. Disco flexible de 5,25"

La operación de lectura se realiza por aplicación de un cabezal lector sobre la superficie del disco mientras éste gira a gran velocidad (300-360 rpm). Dependiendo del tipo de disco y del empleo simultáneo de una o dos caras, los cabezales de grabación y lectura serán simples o dobles. La grabación y lectura se realiza sobre anillos concéntricos denominados pistas (de 35 a 80), las cuales a su vez se subdividen en sectores (de 9 a 26).

En la figura 12.22 se indica la numeración de las pistas, el orden de comienzo de numeración de las mismas y la división en sectores de las pistas, para un ejemplo de los posibles discos flexibles que pueden encontrarse en el mercado. En función de todos estos parámetros, pueden existir los siguientes tipos de discos:

250K (simple densidad) para 8 pulgadas
500K (doble densidad) para 8 pulgadas
360K (doble densidad) para 5,25 pulgadas
1200K (alta densidad) para 5,25 pulgadas

720K (doble densidad) para 3,5 pulgadas
1440K (alta densidad) para 3,5 pulgadas
128K (simple densidad) para 3 pulgadas
256K (doble densidad) para 3 pulgadas

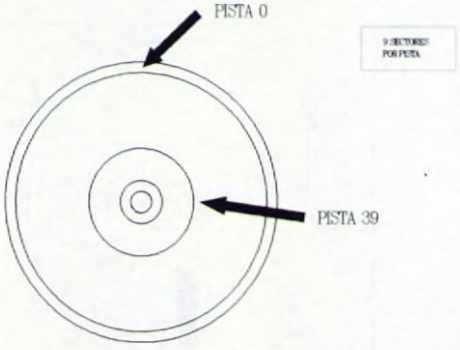


Figura 12.22. Numeración de las pistas

12.8.2. **Cinta magnética**

Es un soporte de información cuya base es una cinta de material plástico recubierta por una de sus caras con material magnetizable. Existen cintas especiales para informática de tamaño, calidad y duración diferentes a las clásicas cintas de cassette, aunque existen gran cantidad de ordenadores personales de bajo coste que usan estas últimas. El formato de estas cintas profesionales puede ser en cartuchos, de tamaño inferior al de un cassette estandar, o en bobinas de diámetro superior a los veinte centímetros. Las cintas magnéticas, que podemos denominar como "profesionales", son de grabación y lectura multipista, lo que obliga a disponer de cabezales adecuados.

Son de muy alta densidad de grabación, superando los 40 millones de caracteres, pero tienen la desventaja respecto al disco, del elevado tiempo de lectura y grabación (varios minutos frente a segundos). En la cinta la información se graba en bloques, separados por bandas que regulan la velocidad de arrastre de la misma. Por su gran capacidad, tiene uso preferencial en la realización de copias de seguridad de la información que se almacena normalmente en discos rígidos y flexibles. (Se denomina "streamer" al dispositivo de grabación en cinta para copias de seguridad)

12.8.3. **Tarjetas magnéticas**

Son trozos de cinta magnética sobre un soporte plástico. Es necesario su introducción manual en los equipos de lectura y grabación y por su peculiaridad y capacidad de almacenamiento no se usan en computadores y están casi restringidas a calculadoras programables y terminales portátiles. En los ordenadores que aún mantienen este sistema, se almacenan y manipulan las tarjetas magnéticas como las tarjetas perforadas, tomándolas de su lugar de almacenamiento y llevándolas a un cilindro de lectura.

12.8.4. **Tambor magnético**

Son los soportes de información de menor tiempo de respuesta. Están formados por un cilindro recubierto de material magnetizable. Gira a muy alta velocidad y lleva montado en la superficie exterior los cabezales de lectura y grabación. Debido a la velocidad de rotación, los cabezales no tocan la superficie. Las pistas del tambor son cilíndricas y paralelas entre sí y por su complicada constitución no son normalizados y no pueden ser transportables, formando un conjunto homogéneo particularizado en muchos casos para cada ordenador.

12.8.5. **Disco rígido**

Denominado hard disk o disco duro, está constituído por un disco metálico rígido de aluminio, con las 2 caras impregnadas de material magnetizable. Según la disposición de los cabezales de lectura y de grabación, pueden encontrarse modelos de cabezales fijos y de cabezales móviles, con características sensiblemente diferentes. Los modelos de cabezales fijos, funcionan como los tambores, con los cabezales en espiral sobre cada cara del disco.

Se realiza un conjunto hermético aislado del polvo y agentes externos, donde se incluye el disco, el motor de rotación y los cabezales. En los modelos de cabezales móviles, se usa un cabezal por cada cara del disco, que va unido a un brazo de movimiento radial. Las pistas del disco son concéntricas y divididas en sectores como en los discos flexibles. Los cabezales tienen perfil aerodinámico, de manera que al girar el disco se produce una corriente de aire que los levanta de la superficie del disco y evita que se produzca rozamiento, con el consiguiente desgaste que ello produciría.

Son los elementos más utilizados como soporte de información auxiliar en los ordenadores actuales, con una capacidad de almacenamiento muy elevada y un tiempo de respuesta muy breve.

TEMA 13
CIRCUITOS PROGRAMABLES

13

CIRCUITOS PROGRAMABLES

13.1. INTRODUCCION

La intención principal del presente tema es profundizar en la programación de las memorias de los sistemas digitales, bien sea por parte del fabricante como por parte del usuario. Para ello, se describirán las memorias RAM y ROM, pasando posteriormente al análisis de las mismas desde el punto de vista de una posible programación.

Como anteriormente se han analizado dichas memorias como bloques funcionales, en este tema se van a desglosar tanto el bloque de entradas, correspondiente al circuito de decodificación de señales de entrada, como el bloque de salida, correspondiente a las conexiones posteriores con los registros de almacenamiento de información de salida.

Estas posibles programaciones de las memorias, definidas en los circuitos más versátiles como aplicaciones lógicas programables, ofrecen múltiples posibilidades de utilización, teniendo en cuenta que además su empleo, análisis y manipulación es sumamente fácil al basarse en la lógica binaria, el álgebra de boole y los circuitos combinacionales. Las aplicaciones prácticas de dichos circuitos digitales se detallan en cada caso mediante ejemplos, ecuaciones y expresiones lógicas y gráficos donde se detallan las conexiones de los mismos.

13.2. CIRCUITOS DIGITALES CON MEMORIAS ROM

Tal como se ha definido anteriormente, las memorias ROM son memorias de acceso aleatorio, no volátiles y sólo de lectura. Esto quiere decir que vienen grabadas de fábrica, o lo que es lo mismo, que el usuario no puede acceder a su conexionado interno y modificar su estructura de funcionamien-

to. Teniendo en cuenta que su estructura básica puede considerarse la misma que la de las memorias RAM, se puede describir un circuito genérico para ambos casos, constituido por una matriz de biestables.

Si se dispone de una estructura matricial como la de la figura 12.14, mediante el decodificador de líneas horizontales, se selecciona una línea, que corresponde a una posición de memoria. Este decodificador es el que se encarga de localizar una palabra, cuya dirección es conocida. Si se analiza por comodidad una sencilla estructura de 16 líneas, se necesita un decodificador de 4 entradas y 16 salidas, y de esta manera, la combinación de los 4 bits de entrada definirá la posición o dirección de memoria que se trata de localizar.

Para n entradas, se obtendrán 2^n salidas del decodificador (posiciones de memoria). Por ejemplo, una memoria con un decodificador de 16 entradas, podrá direccionar 65536 posiciones de memoria, o lo que es lo mismo, 64K posibles palabras de memoria. La longitud de la palabra depende del equipo que se utilice, pero por ejemplo, para una longitud de palabra de 8 bits, la más usual en ordenadores personales, la memoria contendrá un número de bits igual a 64K x 8 = 512K. De esta manera, al direccionar una posición de memoria, que en este caso, al tratarse de una ROM, lo que se realiza es una lectura de la información grabada, se obtiene a la salida de la memoria una palabra de 8 bits correspondiente a la línea seleccionada por el decodificador, sin que se altere el contenido de la memoria. Un empleo típico de la memoria ROM puede ser el de iluminar un display 7 segmentos.

La información necesaria para activar cada segmento del display viene preprogramada de fábrica, siendo necesario tan sólo conectar siete de las

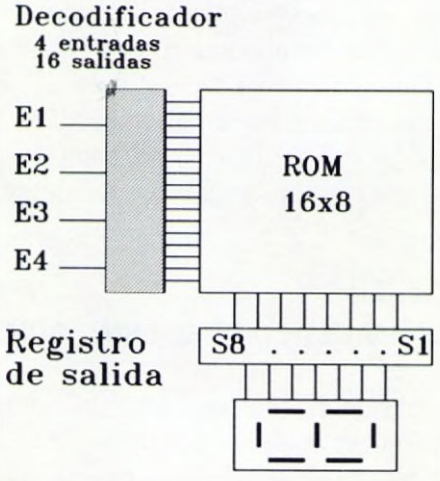


Figura 13.1. Iluminación de un display con una ROM

ocho salidas de la memoria directamente al display siete segmentos. Las salidas S_1 a S_7 de la matriz de memoria se conectarán directamente a los terminales a, b, c, d, e, f, g del display 7 segmentos, o bien a través de un convertidor de código si fuera necesario.

En el caso que se está analizando, con 8 bits de salida, no será necesario conectar el convertidor de código, ya que la señal obtenida a la salida se puede conectar directamente al display siete segmentos y no es necesario decodificarla. Analizando entonces las dieciséis posiciones de memoria que se ocuparán en función del número de combinaciones de las señales de entrada del decodificador de la memoria, se pueden analizar las señales necesarias para obtener la información grabada en cada una de las líneas de la memoria. Al actuar sobre cada una de dichas posiciones de memoria, el resultado obtenido será la información grabada en la misma, que será visible inmediatamente en el display siete segmentos conectado con el registro de salida. La tabla que relaciona entradas y salidas será la misma que la definida para el diseño de un convertidor de código. El esquema de conexiones quedará por tanto según se indica en la figura 13.1. En dicho esquema la conexión S_8 del registro de salida no tendrá utilidad, dado que sólo serán empleadas siete de las mismas, las necesarias para aplicarlas a las entradas del display. (La octava para el punto decimal)

Si se necesita ampliar la capacidad de la memoria a emplear, puede realizarse conectando varias entre sí, como en la figura 13.2, introduciendo un decodificador adicional y varias entradas de selección.

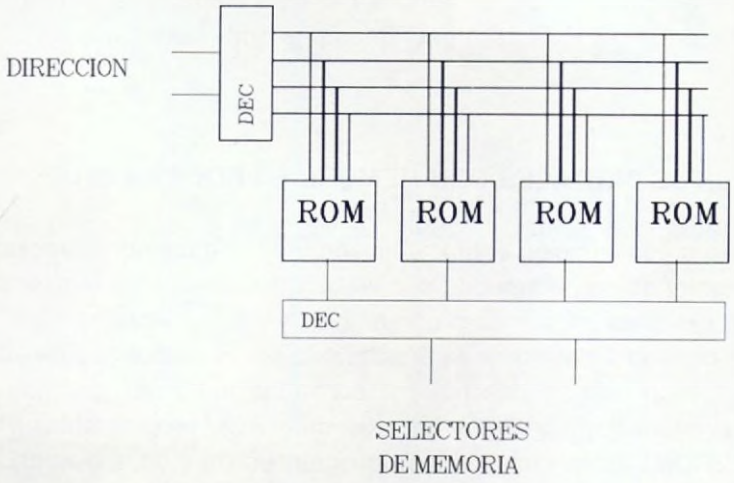


Figura 13.2. Selección de memorias

13.3. CIRCUITOS DIGITALES CON MEMORIAS RAM

Descritas ya anteriormente al igual que las ROM, las memorias RAM o memorias de acceso aleatorio, volátiles y de lectura y escritura, constituyen el almacenamiento de información más versátil e importante del interior de un sistema digital, especialmente de los sistemas microprocesadores, empleándose generalmente como memoria de "trabajo" de dichos sistemas por su posibilidad de ser regrabada continuamente.

Como su estructura básica puede suponerse similar a la de las memorias ROM, seguirá usándose el circuito detallado en el gráfico de la figura 12.14. El análisis realizado para las memorias ROM es totalmente adjudicable también al funcionamiento de las memorias RAM, constituyendo el cincuenta por ciento de sus posibilidades. El otro cincuenta por ciento corresponde a la posibilidad de grabar información en las posiciones de memoria.

Si las combinaciones de las señales de entrada determinaban la posición de memoria donde estaba la información grabada que se quería leer, esas mismas combinaciones de entrada determinan asimismo la posición de memoria donde se puede grabar la información que se desee en cada instante. En función del tipo de estructura interna que posea la memoria, bien mediante la estructura 2D, 2 1/2D o 3D, se determinará la forma de realizar la grabación y la lectura. En ambos casos, lectura y grabación, la memoria empleará el mismo tiempo de ciclo. Por tanto, el caso analizado del display siete segmentos con la memoria ROM puede ser perfectamente aplicado a las memorias RAM, teniendo en cuenta que la información contenida en la memoria habrá que grabarla inicialmente. La tabla del convertidor de código indicada para la formación de la memoria ROM pregrabada de fábrica será la información que se necesita para grabar las posiciones de memoria.

13.4. CIRCUITOS DIGITALES CON MEMORIAS PROGRAMABLES

Se analiza a continuación el funcionamiento de una memoria programable de lectura, teniendo en cuenta que el análisis del circuito será el mismo para los distintos casos de memorias programables y reprogramables. La diferencia entre ellas vendrá determinada únicamente por la posibilidad de volver a repetir el proceso que se describirá a continuación, o por los medios de borrado de la información grabada. Las memorias ROM programables (PROM, EPROM, EEROM) están constituídas por matrices de diodos o transistores, los cuales son seleccionados mediante un dispositivo programador y actúan como fusibles al paso de la corriente eléctrica. Mediante intensidades de alto

valor se consigue que los diodos o transistores seleccionados modifiquen su estructura intrínseca, permitiendo el paso de la corriente eléctrica o bloqueándola según la programación realizada. El funcionamiento de dichos elementos puede hacerse similar al de un fusible, que al paso de una elevada corriente bloquea el resto del circuito.

Sin entrar en análisis electrónicos de las características de dichos semiconductores, se analizan las memorias programables desde el punto de vista de estructuras lógicas digitales formadas por conexiones de puertas lógicas, cuya estructura final es un sistema bloque con varias entradas y salidas. En la figura 13.3 se describe una simplificación esquemática que se realizará en todos los gráficos y esquemas de este apartado y siguientes del presente tema.

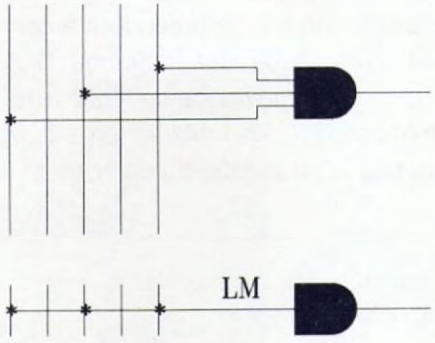


Figura 13.3. Simplificación esquemática

Para evitar confusión en la interpretación de los circuitos, dado el elevado número de líneas y conexiones a realizar, todas las líneas de entrada a una puerta lógica se reducirán a una sola, indicando los puntos de conexión con otras líneas. Analizando la figura 13.3, se observa que la estructura mostrada en el primer gráfico, se simplifica según se indica en el segundo gráfico, reduciendo las 3 líneas de entrada a una sola, con los puntos de conexión sobre la misma y la referencia LM de línea múltiple. Conviene no olvidar que se están analizando memorias de semiconductores de cualquiera de los tipos posibles y que por tanto, cuando por comodidad o simplificación se hace referencia a un punto de conexión en la matriz, se está indicando el diodo o transistor correspondiente a dicha conexión (es decir, no es un cruce o conexión de cables por lo cuál no se puede hablar de conexión eléctrica). Ampliando dicho punto de conexión para su análisis, mostraría la imagen de la figura 13.4 en el caso de que utilizar memorias de estructura matricial formadas por diodos semiconductores o biestables.

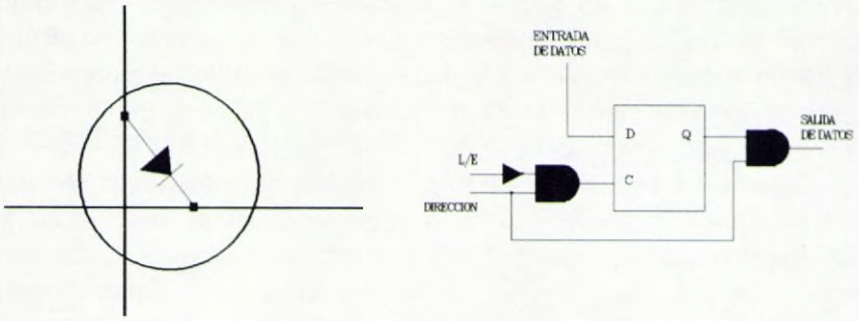


Figura 13.4. Ampliación de una conexión

Teniendo en cuenta este criterio, se procederá a continuación a realizar el circuito de conexiones internas de una memoria ROM programable de 3 entradas, E_1, E_2, E_3, que para mayor comodidad a la hora de realizar los conexionados esquemáticos, se desdoblarán en sus valores complementarios (valores negados), según se puede observar en la figura 13.5.

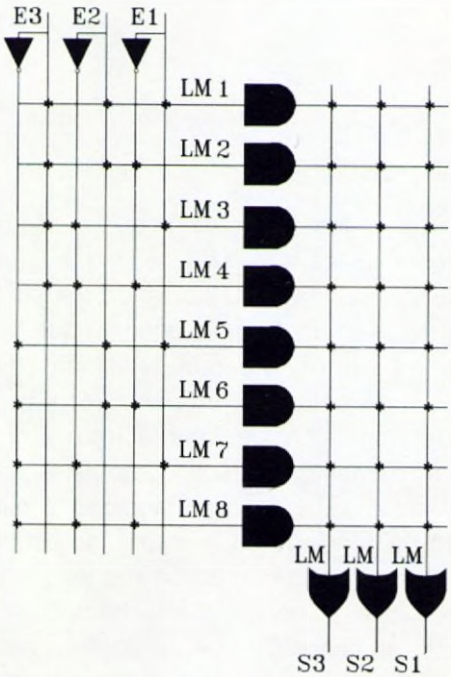


Figura 13.5. Memoria PROM

Los valores de las tres entradas al circuito, darán lugar a ocho posibles combinaciones binarias, por lo que se disponen ocho puertas lógicas AND, que representarán los ocho posibles productos binarios. Al ser una memoria sólo de lectura, los elementos semiconductores (conexiones) de la matriz formada por las entradas y las puertas AND vendrán determinados desde la fabricación y serán fijos e inalterables según las combinaciones binarias equivalentes a las variables de entrada, que para el caso que se analiza con tres variables serán las siguientes:

E_3	E_2	E_1
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Sin embargo, al tratarse de una memoria programable, existe la opción de seleccionar las combinaciones de las salidas de las puertas AND. Para ello se expresan las salidas como suma de productos, por lo que se define otra estructura matricial formada por las salidas de las 8 puertas AND y las entradas a las puertas sumadoras OR que expresarán las salidas. Así y tal como se desprende de la figura 13.5, cada salida estará definida según las expresiones algebraicas que se deseen mediante la combinación lógica de los valores que tengan las diferentes entradas.

Si se utilizan memorias de mayor longitud de palabra, por ejemplo 8 bits, se tienen 8 salidas y 4 entradas, que corresponde a decodificar 16 señales. Su esquema representativo, se muestra en forma de diagrama de bloques en la figura 13.6 y desarrollado su conexionado interno en la figura 13.7.

Se obtiene una estructura interna de 16 líneas y 8 columnas, lo que equivale a 128 bits, cifra de la que resultan múltiplo bastantes memorias estándar. Mediante la agrupación de memorias elementales como la que se ha detallado, se consiguen los valores usuales de capacidad de los equipos digitales.

Figura 13.6. Bloque decodificador de una memoria de 16 líneas

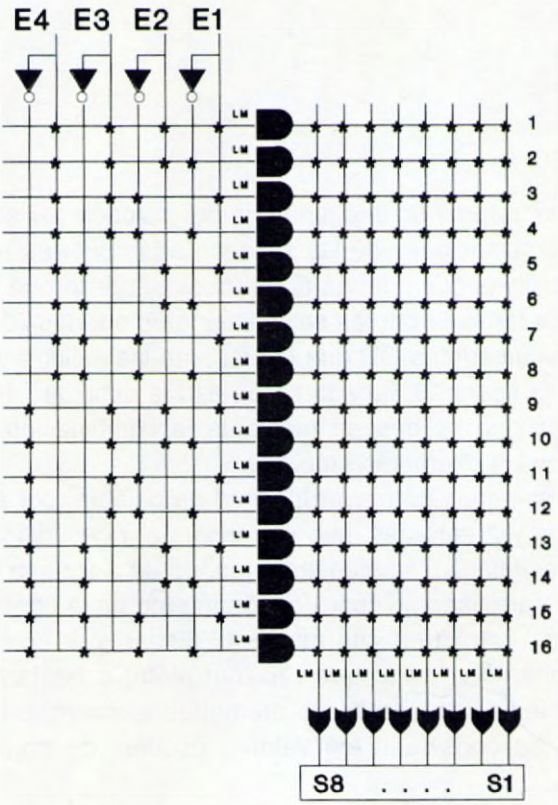


Figura 13.7. Memoria PROM de 16 líneas

*## **Ejemplo 13.1**: Obtener las siguientes expresiones S_1, S_2 y S_3 con una matriz PROM suponiendo una longitud de palabra de 3 bits.*

$$S_1=E_3\bar{E}_2E_1+\bar{E}_3E_2\bar{E}_1+E_3\bar{E}_2\bar{E}_1+\bar{E}_3E_1$$
$$S_2=E_3E_2E_1+E_3\bar{E}_2+E_2\bar{E}_1$$
$$S_3=\bar{E}_3\bar{E}_2+E_3E_2\bar{E}_1$$

Solución:

Para analizar la disposición final del circuito, se formará una tabla para indicar las salidas activas de las puertas lógicas AND y de las puertas lógicas OR, según se detalla en los valores de la tabla siguiente y de la figura 13.8.

Puerta	E_3	E_2	E_1	S_3	S_2	S_1
A_1	0	0	0	1	0	0
A_2	0	0	1	1	0	1
A_3	0	1	0	0	1	1
A_4	0	1	1	0	0	1
A_5	1	0	0	0	1	1
A_6	1	0	1	0	1	1
A_7	1	1	0	1	1	0
A_8	1	1	1	0	1	0

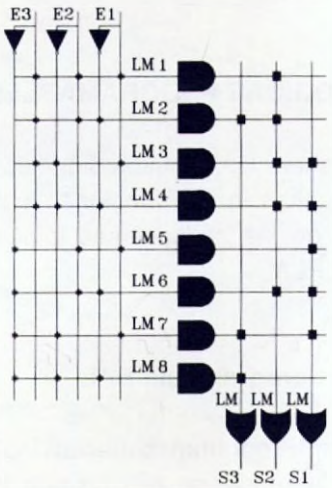


Figura 13.8

*## **Ejemplo 13.2**: Aplicar el circuito de la memoria PROM a la iluminación de un display 7 segmentos.*

Solución:

Puerta	E_4	E_3	E_2	E_1	S_8	S_7	S_6	S_5	S_4	S_3	S_2	S_1
A_1	0	0	0	0	-	0	1	1	1	1	1	1
A_2	0	0	0	1	-	0	0	0	0	1	1	0
A_3	0	0	1	0	-	1	0	1	1	0	1	1
A_4	0	0	1	1	-	1	0	0	1	1	1	1
A_5	0	1	0	0	-	1	1	0	0	1	1	0
A_6	0	1	0	1	-	1	1	0	1	1	0	1
A_7	0	1	1	0	-	1	1	1	1	1	0	0
A_8	0	1	1	1	-	0	0	0	0	1	1	1
A_9	1	0	0	0	-	1	1	1	1	1	1	1
A_{10}	1	0	0	1	-	1	1	0	0	1	1	1
A_{11}	1	0	1	0	-	1	0	1	1	0	0	0
A_{12}	1	0	1	1	-	1	0	0	1	1	0	0
A_{13}	1	1	0	0	-	1	1	0	0	0	1	0
A_{14}	1	1	0	1	-	1	1	0	1	0	0	1
A_{15}	1	1	1	0	-	1	1	1	1	0	0	0
A_{16}	1	1	1	1	-	0	0	0	0	0	0	0

Las salidas $S_1,...,S_7$ del esquema de la figura 13.7 se conectarán respectivamente a los terminales a, b, c, d, e, f, g, del display 7 segmentos, quedando la salida S_8 inhabilitada.

13.5. APLICACIONES LOGICAS PROGRAMABLES

Se definen de esta manera los circuitos digitales similares a las memorias PROM que permiten la realización de funciones lógicas. Se pueden determinar 3 subgrupos dentro de las aplicaciones lógicas programables, que se denominan PAL, PLA y FPLA.

13.5.1. Aplicación lógica programable PAL

Se denomina en inglés "Programmable array logic", y tiene una estructura similar a la PROM. Se diferencia de ella en que la matriz formada por las entradas y las puertas AND (decodificador) es modificable según el uso que

vaya a tener el circuito, pero la matriz de salida de las puertas AND y entradas a las puertas OR es siempre la misma e inalterable. Su esquema se representa en la figura 13.9.

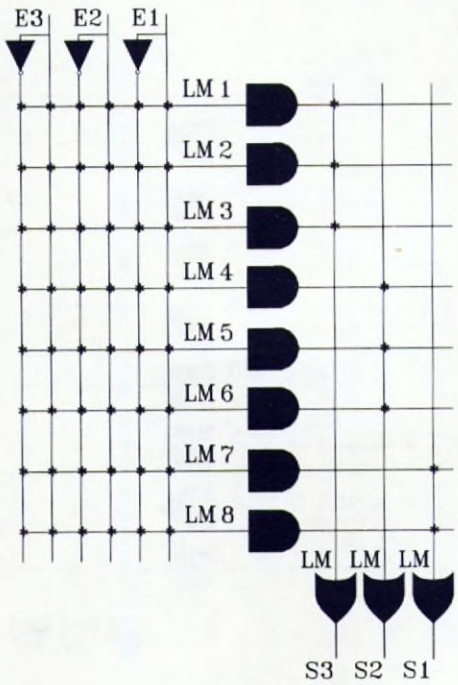


Figura 13.9. Estructura PAL

13.5.2. **Aplicación lógica programable PLA**

Denominada en inglés "Programmable logic array", su estructura es definible completamente. Quiere decir que las 2 matrices descritas como estructura interna pueden ser determinadas según las necesidades del circuito. La definición de la misma es equivalente a la realizada para las otras matrices anteriores, las programables ROM y la aplicación PAL. Su esquema se representa en la figura 13.10.

La gran ventaja de este tipo de aplicaciones, es que la parte del circuito no definida se puede emplear en otras combinaciones o dejarlas inhabilitadas. El coste de estas aplicaciones es mucho menor que el de las memorias programables ROM y las posibilidades de las mismas resultan

mucho mayores dada la gran versatilidad que poseen. Pueden por tanto manejar mayor número de entradas y necesitan menos conexiones y menos elementos para obtener la misma salida, dado que pueden simplificarse las expresiones lógicas.

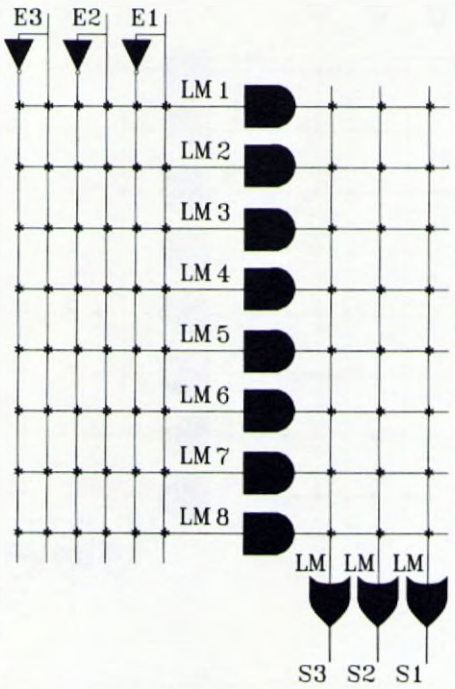


Figura 13.10. Estructura PLA

*## **Ejemplo 13.3**: Obtener el montaje con una PLA de la función lógica siguiente:*

$$S=E_3E_2E_1+E_3\bar{E}_2E_1+\bar{E}_3E_2\bar{E}_1$$

Solución:

Se realiza la tabla de conexiones entre entrada y salida. Se emplean sólo 3 puertas AND y 1 salida para representar la ecuación lógica definida. Quedan disponibles por tanto 5 puertas lógicas AND (equivalentes a otras tantas combinaciones de las entradas), y 2 salidas.

E3	E2	E1	Puerta	S3	S2	S1
1	1	1	A1	-	-	1
1	0	1	A2	-	-	1
0	1	0	A3	-	-	1
-	-	-	A4	-	-	-
-	-	-	A5	-	-	-
-	-	-	A6	-	-	-
-	-	-	A7	-	-	-
-	-	-	A8	-	-	-

El circuito correspondiente se representa en la figura 13.11.

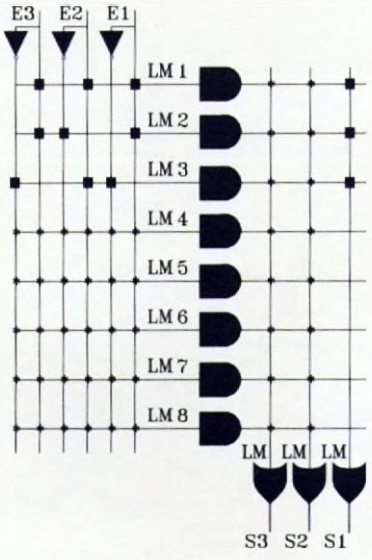


Figura 13.11

13.5.3. **Aplicaciones lógicas programables FPLA**

Corresponden a la versión de las PLA con posibilidad de ser programadas por el usuario (Field programmable logic array). Le son de aplicación todos los comentarios realizados para las PLA, con la ventaja de los posibles cambios de estructura. Al igual que las PLA su coste es mucho menor que las PROM y las posibilidades mucho mayores, dada la gran versatilidad de las mismas.

Asimismo, pueden por tanto manejar mayor número de entradas y necesitan menos conexiones y elementos para obtener la misma salida ya que pueden simplificarse las expresiones lógicas. El campo de las aplicaciones lógicas programables es cada vez más amplio puesto que las memorias y elementos programables y reprogramables están cada vez más al servicio de los usuarios de sistemas digitales, pasando de antiguos equipos rígidos a modernos equipos flexibles donde el programador interviene con más posibilidades.

TEMA 14
PROCESADORES DIGITALES

14

PROCESADORES DIGITALES

14.1. INTRODUCCION

El presente tema ofrece brevemente la descripción interna de un sistema microprocesador, dando unos conocimientos básicos sobre el sistema digital más completo, donde se aplican todos los conceptos detallados en los temas anteriores. Con esta breve introducción se pretende dar a conocer los elementos existentes en el interior de estos equipos digitales tan complejos y sofisticados.

Una vez estudiados los sistemas binarios y hexadecimales, el álgebra de boole, los decodificadores, biestables, memorias, registros, etc., es interesante saber qué ocurre al conectar los diversos elementos entre sí, y cómo se aplican las reglas aprendidas. Por ello, y como forma más fácil de orientar el aprendizaje de dichas conexiones, se ha diseñado un pequeño sistema digital microprocesador que se ha denominado SD-1 para abreviar su expresión al nombrarlo. Sobre el SD-1, se analizarán las conexiones indicadas, el almacenamiento de información, el movimiento de datos, las estructuras clásicas de buses ya conocidas, etc.

Lógicamente al ser simplemente un pequeño esquema de aprendizaje de los sistemas microprocesadores, una vez estudiado a fondo y analizadas y comprendidas las distintas partes de que consta, probablemente a algunas personas les parezca insuficiente, señal de que es importante el interés que sienten por la arquitectura de sistemas informáticos, pudiendo ampliar conocimientos a través de libros de estructura de sistemas y arquitectura de computadores que abundan mucho actualmente dado el gran auge que está teniendo el mercado informático con continuas apariciones de equipos que mejoran la estructura y posibilidades de sus precedentes.

14.2. ESTRUCTURA DE UN SISTEMA MICROPROCESADOR

Independientemente de la diversidad de elementos accesorios de que puede constar un sistema digital y especialmente los sistemas microprocesados, podemos definir una serie de elementos básicos comunes a la totalidad de sistemas existentes en la actualidad. Estos elementos básicos se van a expresar en forma de agrupaciones o bloques de elementos que reciben las siguientes denominaciones:

- UNIDAD DE CONTROL
- UNIDAD DE PROCESO
- MEMORIA
- UNIDAD DE ENTRADAS Y SALIDAS

Estos 4 bloques pueden estar conectados entre sí y al mismo tiempo conectados a otros elementos del sistema. Dichas conexiones se realizarán a través de una estructura de buses que facilitará el conexionado de los mismos, aunque tendrá los inconvenientes mencionados anteriormente para las conexiones en buses o calles. Existen 4 buses con las siguientes denominaciones, que se conectan unidireccional y bidireccionalmente con los bloques según la figura 14.1:

- BUS DE DIRECCIONES
- BUS DE DATOS
- BUS DE CONTROL
- BUS DE COMUNICACIONES EXTERIORES

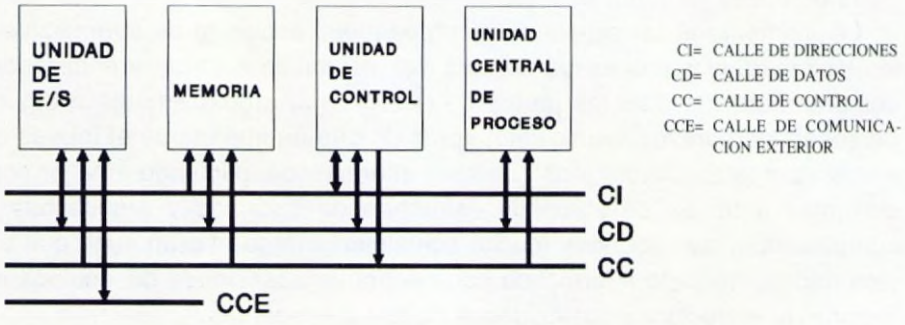


Figura 14.1. Diagrama de bloques de un procesador

14.2.1. **La Unidad de Control**

La Unidad de Control es el elemento del sistema encargado de controlar y gobernar el sistema digital al completo. Decide en cada instante los elementos que han de ser activados y desactivados, genera señales periódicas de referencia, permite el trasvase de información entre los distintos elementos del sistema y autoriza la transferencia de información con el exterior. Sin importar la operación que se esté realizando ni los elementos que intervengan, la Unidad de Control es quien regula y coordina el funcionamiento del conjunto.

14.2.2. **La Unidad de Proceso**

Es el bloque operativo y de cálculo dentro del sistema digital. Su misión es la de realizar las operaciones tanto aritméticas como lógicas que necesite el sistema. Para ello, su elemento básico es la ALU o Unidad Aritmético Lógica, que puede realizar operaciones de suma, resta, multiplicación, división, complementación, operaciones lógicas de suma, producto, equivalencia, etc. Es también en algunos casos almacenador de resultados, especialmente para datos parciales, para lo cuál dispone de registros de almacenamiento temporal de información como el acumulador. Su unión con la Unidad de Control recibe el nombre de unidad central de proceso o microprocesador si es un solo circuito integrado.

14.2.3. **La Memoria**

Su constitución es la descrita en el tema 12. En un sistema digital corresponde esta denominación al bloque que almacena información bien sea temporal o indefinidamente. En este ejemplo a tratar se considerará la memoria como un bloque único, aunque en los sistemas más complejos se puede desglosar en varios subgrupos según sea de almacenamiento temporal (Memoria de trabajo), indefinido (Memoria ROM), para acelerar el sistema (Memoria caché), para comunicarse con el exterior (Memorias tampón o buffers), etc.

La memoria asignará una referencia a cada elemento que almacene para su posterior localización. El almacenamiento se realiza en lo que se denominan posiciones de memoria, siendo cada una de estas posiciones de memoria la referencia que se le asigna a la información contenida. El sistema según sus necesidades realizará el almacenamiento de información en bloques separados para datos o para instrucciones de programa.

14.2.4. **La Unidad de Entradas y Salidas**

Esta es la denominación que se le da al bloque que realiza las comunicaciones con el exterior del sistema digital. Las comunicaciones bien pueden ser de entrada como de salida de datos, por lo que debido al particular uso y denominación de los elementos exteriores al sistema se puede decir que estas comunicaciones las realiza con periféricos, siendo esta la expresión que se da a los elementos externos al sistema digital básico.

Esta unidad debe adaptar entre sí las señales de los distintos periféricos y elementos conectados a ella con el sistema digital microprocesador, ya que normalmente cada producto y cada fabricante tienen determinadas peculiaridades que pueden impedir el correcto trasvase de información si no se tienen en cuenta.

14.2.5. **Los Buses**

Constituyen los elementos de interconexión de los bloques, así como el camino de movimiento de la información en el interior del sistema. Aunque básicamente un bus ó calle es simplemente el "cable" que sirve de conexión a dos o más elementos, reciben denominaciones particulares en base al tipo de elementos a los que sirve de conexión, a la información que transmiten, o a la función que con ella se realice:

- El Bus de Direcciones sirve para llevar a la Memoria del sistema la posición de memoria o dirección donde se ha de grabar la información o donde se encuentra ya grabada y se debe proceder a leerla.

- El Bus de Datos sirve para llevar los datos de la memoria a los diversos elementos que los puedan procesar o utilizar, o para enviar información a la memoria para su grabación.

- El Bus de Control es el que transmite las órdenes de la Unidad de Control a todos los elementos del sistema, por lo que es más difícil apreciarlo como un conjunto de conductores cuando se observa la electrónica interior de un sistema informático.

- El Bus de Comunicaciones Exteriores es el que pone en contacto el sistema microprocesador y el mundo exterior. A través de él recibe información y a través de él la envía a los diversos elementos que tenga conectados exteriormente el sistema informático.

14.3. EJEMPLO DE UN SISTEMA PROCESADOR. EL SD-1

Para diseñar el sistema SD-1, partiremos de la definición previa de cada uno de los bloques que lo constituyen, lo que servirá al mismo tiempo para ir describiendo las funciones y el conexionado de los elementos constituyentes.

La Unidad de Control tiene como elemento central el circuito de control, formado por un circuito combinacional y que es quien emite las señales de control. Este circuito combinacional para su funcionamiento, necesita recibir señales de los demás elementos del sistema, así como señales externas de gobierno. Dichas señales las recibe a través de un decodificador y de un bloque generador de señales periódicas formado por unos elementos denominados Reloj y Generador de secuencias.

Estos elementos son puestos en funcionamiento mediante una señal externa de puesta en marcha, así como también son detenidos con una señal externa de parada. Otros elementos importantes de la unidad de control serían los registros denominados Contador de Programa y Registro de Instrucciones. El contador de programas indica la posición de memoria que debe enviarse a la Memoria bien para leer o para grabar. El registro de instrucciones almacena las instrucciones de programa procedentes de la memoria.

La Unidad de Proceso tendrá como base una ALU de 8 operaciones para no constituir un sistema muy complejo. Estas operaciones pueden ser aritméticas y lógicas, de forma equivalente al operador combinacional descrito en el tema 8. Para seleccionar la operación que la ALU ha de realizar, existen 3 selectores de operación que estarán gobernados desde la unidad de control. Estos tres selectores, mediante las ocho posibles combinaciones binarias que pueden tomar, decidirán la operación a efectuar.

Como todo bloque básico ALU, será necesario incorporarle un registro acumulador de resultados de las operaciones y un registro de estado donde se indique si existe acarreo "C" (Carry), si el resultado es negativo "N" (Negative), si el resultado es cero "Z" (Zero), si existe rebosamiento en la operación "O" (Overflow) y si existe error "E" (Error).

La Memoria está constituida por el bloque matricial de almacenamiento con longitudes de palabra de 8 bits, donde se almacenan las instrucciones y datos, y por unos registros auxiliares para comunicación y transmisión de información. Estos registros se denominan de direcciones y de memoria. El registro de direcciones sirve de comunicación con la calle de direcciones, por donde recibe la petición de la posición de memoria o dirección que se va a emplear. El registro de memoria comunica este bloque con la calle de datos a donde envía información de la posición de memoria pedida y de donde recibe información que ha de ser grabada en la memoria.

La Unidad de Entradas y Salidas está constituida por una serie de elementos, siendo su bloque central el denominado Unidad de Intercambio, constituido por elementos multiplexores que relacionan varias señales simultáneas con una sola línea de transmisión mediante códigos selectores. La unidad de intercambio utiliza un conjunto de señales denominadas Interfases (Igual denominación que el cable físico que conecta los sistemas) para conectar con los elementos exteriores denominados periféricos.

Como se ha indicado anteriormente, las señales de los periféricos y la de los sistemas centrales suelen ser diferentes según los fabricantes de equipos, por lo que ambos se conectan a través de unos elementos denominados Controladores, que adaptan las señales externas e internas. La unidad de intercambio necesita recibir información a transmitir al exterior, así como debe enviar al sistema la información que se le suministra desde el exterior. Esto se realiza a través de la calle o bus de datos.

En la figura 14.2 se representa el esquema completo del circuito SD-1. Se simboliza el bus de control mediante la agrupación de las señales de control asignadas a cada triestado de conexión. Cada triestado conecta o desconecta a los elementos correspondientes según las señales que le lleguen de la unidad de control. Se establece como norma de expresión de dichas señales de control al conjunto formado por el elemento de donde parte la señal y el elemento donde termina dicha señal. Así por ejemplo, la señal de control que activará la línea de unión entre el acumulador AC y la calle de datos CD se expresará como AC-CD. En los casos en que no exista unión entre 2 elementos se representará la orden correspondiente, como en el caso de ciclo de lectura que se representa con una L. A continuación se detallan todas las expresiones que aparecen en el SD-1:

AC=Registro acumulador
ALU=Unidad aritmético-lógica
RE=Registro de estado
CP=Contador de programa
RI=Registro de instrucciones
RD=Registro de direcciones
CI=Calle de direcciones
M=Memoria
RM=Registro de memoria
CD=Calle de datos
DEC=Decodificador
C.C.=Circuito de control
R=Reloj
GS=Generador de secuencias
CC=Calle de control
M=Pulsador de puesta en marcha
P=Pulsador de parada
U.I.=Unidad de intercambio
INT=Interfase
C_1=Controlador
C_2=Controlador
P_1=Periférico
P_2=Periférico
ALU-AC=Conecta ALU y AC
AC-ALU=Conecta AC y ALU
AC-CD=Conecta AC y CD
CD-ALU=Conecta CD y ALU
CI-CP=Conecta CI y CP

CP-CI=Conecta CP y CI
ICP=Incrementa el registro contador de programa
RI-CD=Conecta RI y CD
CD-RI=Conecta CD y RI
CP-CI=Conecta CP y CI
RI-CI=Conecta RI y CI
CI-RD=Conecta CI y RD
CD-CI=Conecta CD y CI
L=Ejecuta un ciclo de lectura de la memoria
E=Ejecuta un ciclo de escritura de la memoria
CD-RM=Conecta CD y RM
RM-CD=Conecta RM y CD
RI-UI=Conecta RI y UI
CD-UI=Conecta CD y UI
UI-CD=Conecta UI y CD
Selectores=Seleccionan la operación de la ALU

14.4. **FUNCIONAMIENTO DEL SD-1**

Suponiendo la señal de puesta en marcha del sistema ha sido accionada, y que por tanto el conjunto reloj-generador de secuencias está en funcionamiento, lo primero que se puede apreciar es que la unidad de control entra en acción y que por tanto el circuito de control empieza a activar sus señales de control para gobernar el sistema completo. En función de las necesidades de uso del sistema, bien sea para archivar información, leer datos almacenados en memoria, procesar datos, etc., el sistema empleará los distintos elementos que lo componen.

Se supone que existe en la memoria del sistema un programa que ha sido grabado previamente. Para poder emplearlo será necesario proceder a leer las instrucciones de que consta. Para ello en el contador de programa CP se dispondrá la posición de memoria o dirección de la primera instrucción del citado programa, transmitiéndose dicha posición a la calle de direcciones CI a través de la señal de control CP-CI. Una vez en CI se transmitirá al registro de direcciones RD a través de la señal de control CI-RD. El registro de direcciones transmitirá a la memoria la dirección buscada donde se encuentra la instrucción I que se desea leer, la cuál mediante un ciclo de lectura L coordinado por el circuito de control, será almacenada temporalmente en el registro de memoria RM.

Del registro de memoria, mediante la señal de control RM-CD será transmitida la instrucción a la calle de datos CD y de aquí mediante la señal de control CD-RI se almacenará en el registro de instrucciones RI. Con la información almacenada en RI momentáneamente detendremos el proceso en lo que vamos a denominar etapa 1, para analizar algunos detalles importantes. El esquema de los pasos realizados en la etapa 1 se muestra en la figura 14.3.

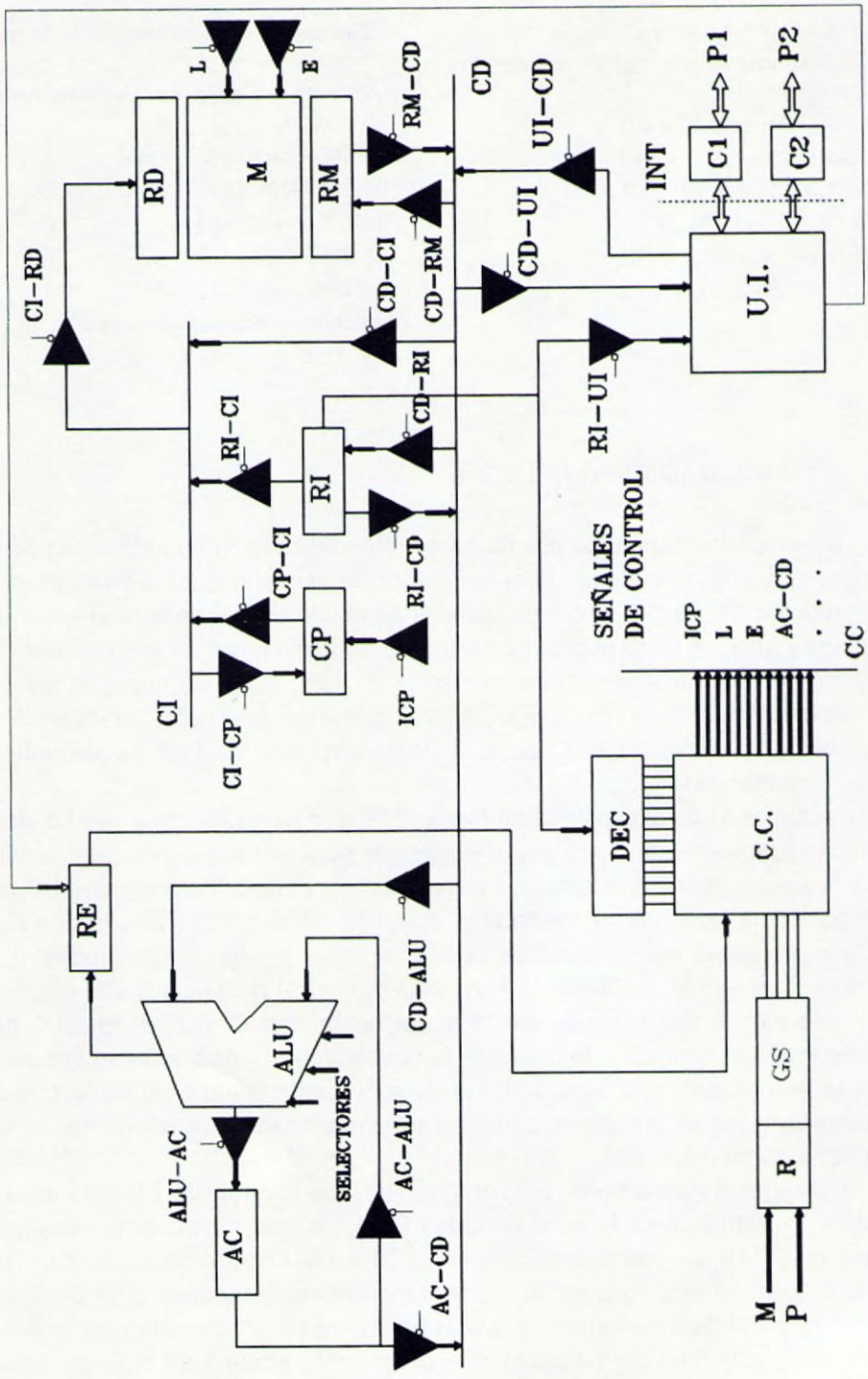


Figura 14.2. El sistema digital SD-1

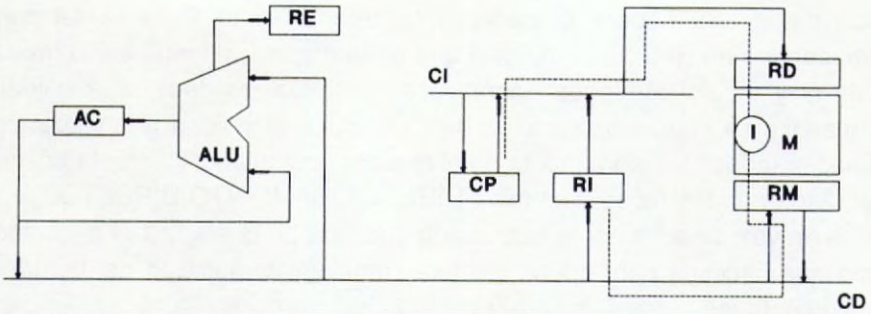


Figura 14.3. Etapa 1

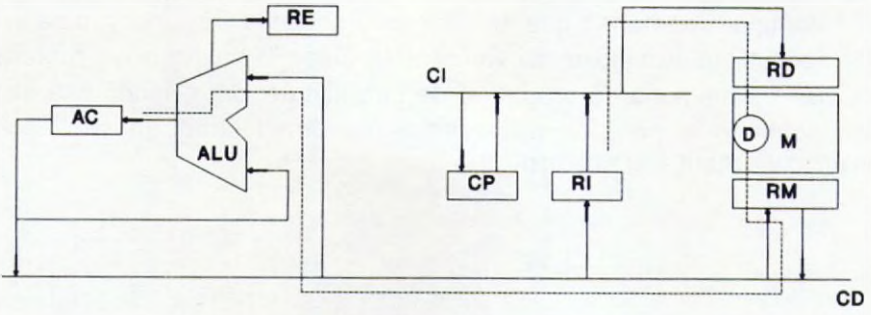


Figura 14.4. Etapa 2

Cuando se hace mención a la actuación de una señal de control se supondrá que en ese instante es la única señal que está actuando, con lo cuál se evitará la posible confusión en caso de que una determinada instrucción pudiera tomar varios caminos de actuación o almacenamiento. (En la realidad las señales se pueden simultanear dependiendo del período de la señal de reloj y de las distintas secuencias)

El estudio de tiempos se realiza mediante cronogramas, en los cuáles se indica la duración de la señal de reloj, las secuencias generadas y los tiempos de actuación de las señales de control. La etapa 2 comienza con una instrucción de programa en el registro de instrucciones RI. La instrucción almacenada en RI puede ser de varias formas, es decir, puede contener información sobre otra instrucción, puede contener información sobre un dato, puede contener el dato, etc.

Si contiene información sobre la posición de memoria donde se encuentra el dato que necesita para seguir adelante, dato con el que realizará alguna operación, esta dirección pasaría a CI a través de la señal de control RI-CI, pasando posteriormente a RD a través de CI-RD. Se ejecutaría un nuevo

ciclo de lectura L para proceder a capturar el dato D de la memoria y almacenarlo en RM. De aquí, igual que en la etapa 1, pasaría a CD mediante RM-CD y de ahí al bloque operativo del sistema, es decir, a la unidad de proceso y en concreto, a través de CD-ALU a la unidad aritmético lógica, almacenándose temporalmente en el registro acumulador AC. Esta búsqueda del dato en memoria se denomina DIRECCIONAMIENTO DIRECTO.

Si en vez de contener la instrucción buscada en la etapa 1 la dirección del dato que necesita para operar hubiera contenido la posición de memoria de la siguiente instrucción, se realizaría un ciclo parecido, salvo que en vez de leer un dato de la memoria, se realizaría la lectura de una nueva instrucción pasando posteriormente de la calle de datos al registro de instrucciones RI nuevamente.

A continuación habría que analizar esta segunda instrucción para ver de qué tipo es, realizándose nuevamente la etapa 2 en caso de contener la posición de memoria de un dato o de otra instrucción. Cuando una instrucción contiene la posición de memoria de otra instrucción se denomina DIRECCIONAMIENTO INDIRECTO.

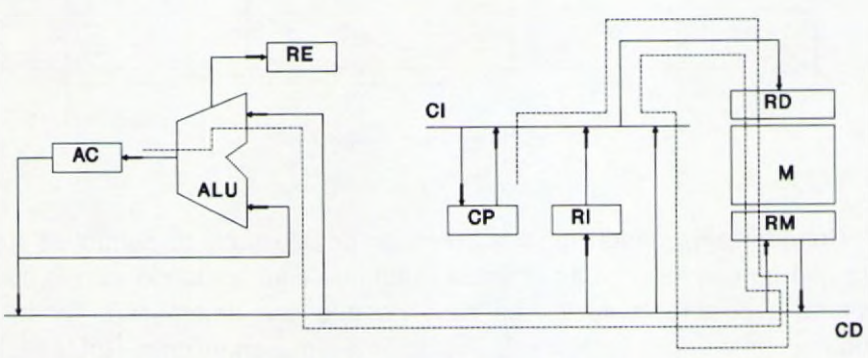


Figura 14.5. Direccionamiento indirecto

El direccionamiento indirecto corresponde a la posibilidad de buscar una instrucción de memoria cuya posición viene indicada en la instrucción que se ejecuta. Es lo que se denomina puntero. Para realizar este ciclo de búsqueda rápida de información se necesitan 2 ciclos de lectura, pero no se necesita pasar por el registro RI, sino que se habilita un paso directo de información entre las calles CD y CI, controlado por la señal CD-CI.

Se denomina DIRECCIONAMIENTO INMEDIATO a un tipo de instrucción que no necesita realizar la etapa 2 para obtener el dato pedido por el programa. Este tipo de instrucciones lleva el dato como parte de la instrucción, por lo cuál, del registro de instrucción RI donde se ha almacenado temporalmen-

te la instrucción que se está ejecutando, se envía el dato a la calle de datos CD mediante RI-CD, pasando a continuación a la ALU mediante CD-ALU.

Una vez el dato en AC se puede realizar la etapa 3 pendiente de ser procesado por la ALU. Mediante los selectores se programa la ALU para realizar la operación necesaria. Cada sistema informático puede procesar los datos mediante un número distinto de operaciones, lo que puede redundar en potencia de cálculo y rapidez. Si suponemos que este sencillo sistema puede disponer de 8 operaciones diferentes de la ALU a seleccionar mediante 3 selectores, las combinaciones binarias de los 3 bits de selección determinan la operación a realizar. Así por ejemplo, si las operaciones a realizar pueden ser la suma, resta, complementación, etc., el código "010" podría significar la operación de complementación.

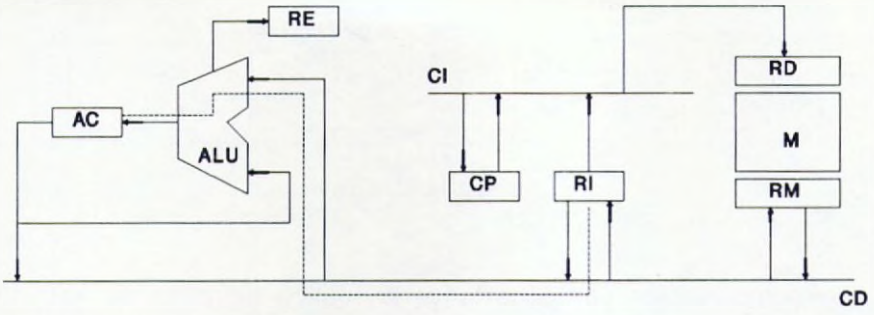


Figura 14.6. Direccionamiento inmediato

La información del registro AC será utilizada por la ALU, conectándose a su segunda entrada a través de la señal de control AC-ALU, almacenándose en AC los resultados de las operaciones parciales y el resultado final, tal como se indica en el esquema de la figura 14.7.

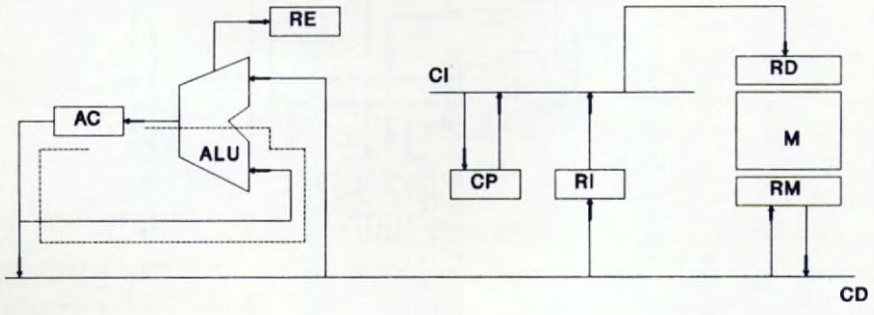


Figura 14.7. Etapa 3

Una vez terminada la etapa de cálculo operativo se comienza la etapa 4, que corresponderá al almacenamiento definitivo del resultado obtenido para una posterior utilización en nuevos cálculos o como resultado final del programa. En esta etapa 4, el dato almacenado en AC pasará a la calle de datos CD a través de la señal de control AC-CD y de esta calle, al registro de memoria RM mediante CD-RM. La actuación final del sistema será la de ejecutar un ciclo de escritura E en la memoria, que realizará la grabación de dicho valor en la posición de memoria que le corresponda.

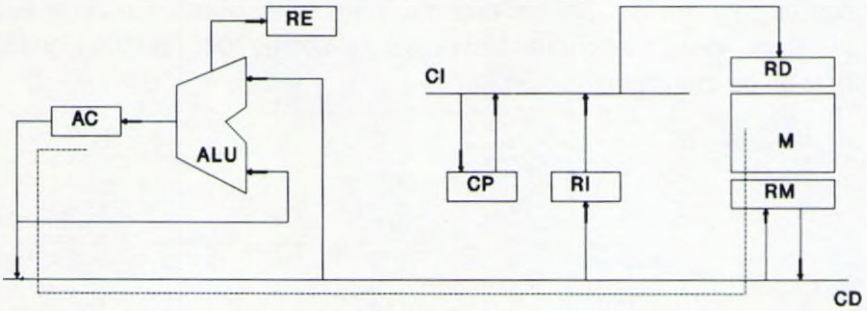


Figura 14.8. Etapa 4

La comunicación con el exterior la realiza a través de estructuras multiplexadas o multicalle. El conjunto de las señales que se envían o reciben del controlador se denominan señales de Interfase (direcciones, datos y señales de control) y sus estructuras responden a los esquemas 14.9 y 14.10.

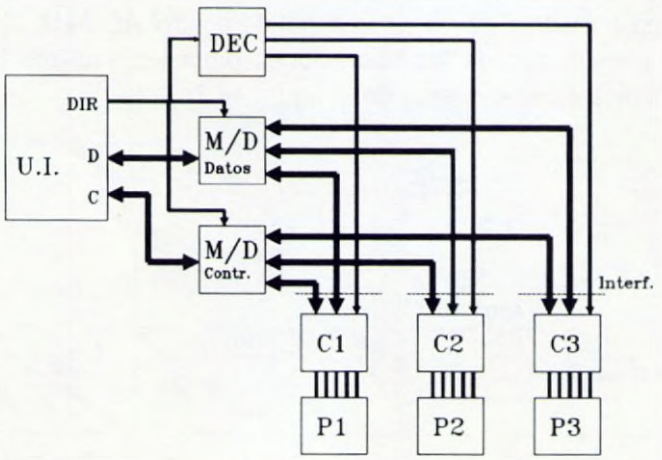


Figura 14.9. Conexión multiplexada

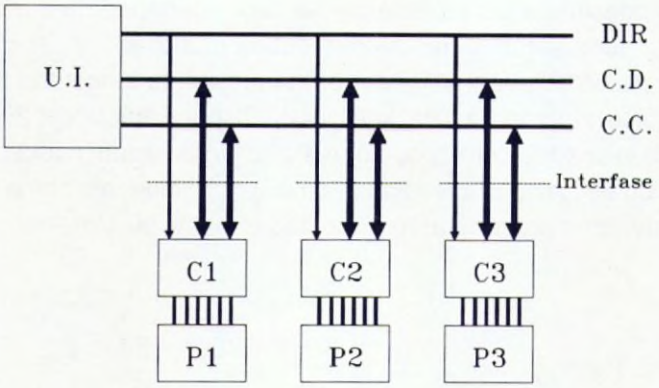


Figura 14.10. Conexión multicalle

Las señales pueden ser transmitidas en modo serie o en paralelo. El intercambio en modo paralelo como la figura 14.11 determina el envío simultáneo normalmente de 8 ó 16 bits. El equipo emisor envía una señal de aviso de comienzo de transmisión (Orden) que es contestada por el controlador del periférico cuando éste este listo para la transmisión (Respuesta). A partir de ese momento puede comenzar la transmisión (E/S=0 para entrada y E/S=1 para salida).

Figura 14.11. Señales de interfase

La transmisión serie es más lenta y responde a estándares algo complejos como el conocido RS-232-C (C equivale a tercera revisión). Esta norma de la EIA (Asociación de Industrias Electrónicas) se diseñó para su empleo en modems, pero su generalización a la conexión de gran variedad de equipos electrónicos e informáticos ha roto la norma, lo que ocasiona que

en muchas ocasiones los terminales del cable genérico de conexión hayan de ser cambiados según el producto y el fabricante.

La transmisión con este estandar es normalmente asíncrona aunque puede ser también síncrona a alta velocidad. Dispone hasta de 25 terminales, siendo los básicos los correspondientes a la transmisión y recepción de datos, la petición de envío y la respuesta de disponibilidad, así como la sincronía. Normalmente las masas estarán conectadas entre sí, existiendo posibilidad

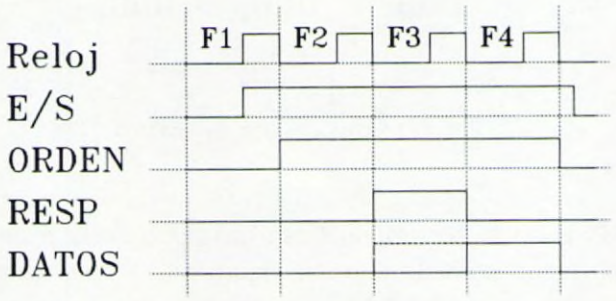


Figura 14.12. Cronograma de intercambio de datos

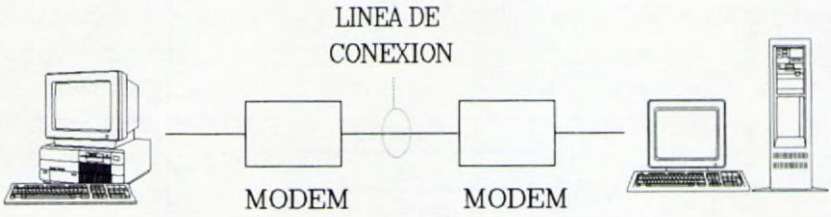


Figura 14.13. Conexión de equipos mediante modem

Figura 14.14. Transmisión serie

de un envío secundario, de señales de confirmación de envío y recepción, de detección de calidad de la señal, indicador de anillo y selectores de velocidad. El equipo transmisor lleva las siglas DTE y el receptor DCE.

La comunicación puede ser SIMPLEX para unidireccional, HALF-DUPLEX para bidireccional no simultánea y FULL-DUPLEX para bidireccional simultánea. Un bit indica el comienzo de la transmisión de datos, normalmente 8 bits pudiendo incluir un bit de paridad, y termina con uno o dos bits de parada.

Figura 14.15. Bits de transmisión serie

14.5. **CICLOS DE FUNCIONAMIENTO**

En función de los elementos que intervienen en la realización de una determinada operación, se invertirá un tiempo diferente de ejecución de la misma. Como el sistema necesita estar sincronizado además de controlado, se dispone de una señal de reloj patrón de referencia y control del conjunto. Si al mismo tiempo cada una de las etapas detalladas anteriormente se hacen coincidir en duración con uno o más ciclos de la señal de reloj, podemos constituir lo que se denomina generador de secuencias.

El generador de secuencias será el elemento clave para poder optimizar los ciclos de funcionamiento de cada una de las etapas del sistema microprocesador. Dicho generador puede facilitar en cualquier momento el funcionamiento simultáneo de varios elementos del sistema si sus fases de trabajo no son incompatibles. Si el SD-1 dispone de 4 etapas de funcionamiento, se pueden hacer equivalentes a 4 secuencias diferentes de duración una señal de reloj cada una. En la figura 14.16 se representa un posible circuito generador de secuencias basado en un contador de anillo y en la 14.17 un registro de desplazamiento que realiza la misma función.

El cronograma esquemático que representa el funcionamiento de ambos elementos se representa en la figura 14.19, donde además del ciclo normal de funcionamiento se indican el ciclo corto empleando direccionamiento inmediato y un ciclo largo de 5 secuencias que se solapa con el comienzo de un nuevo ciclo, sobre todo en operaciones aritméticas de larga duración.

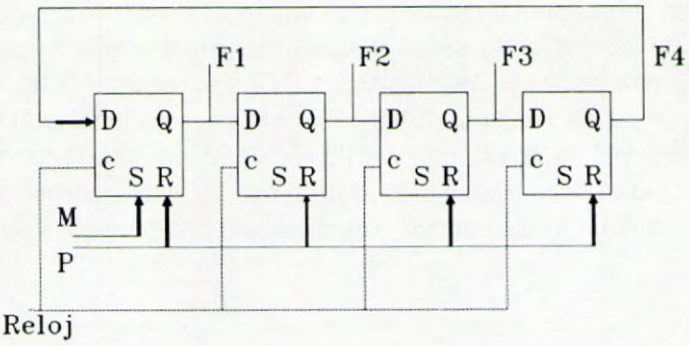


Figura 14.16. Reloj y generador de secuencias

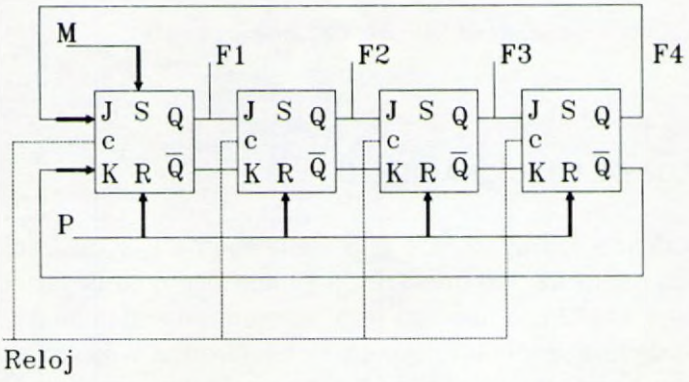


Figura 14.17. Reloj y generador de secuencias (Mod. 2)

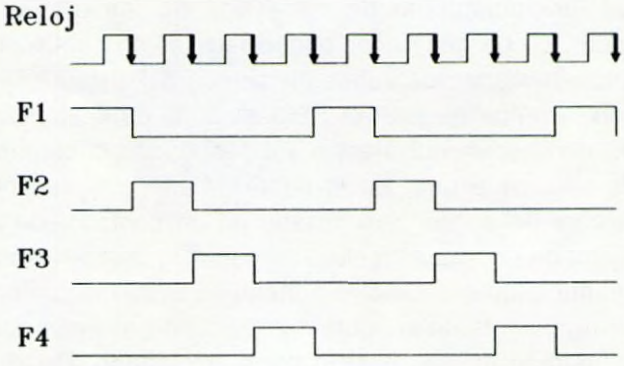


Figura 14.18. Cronograma de secuencias

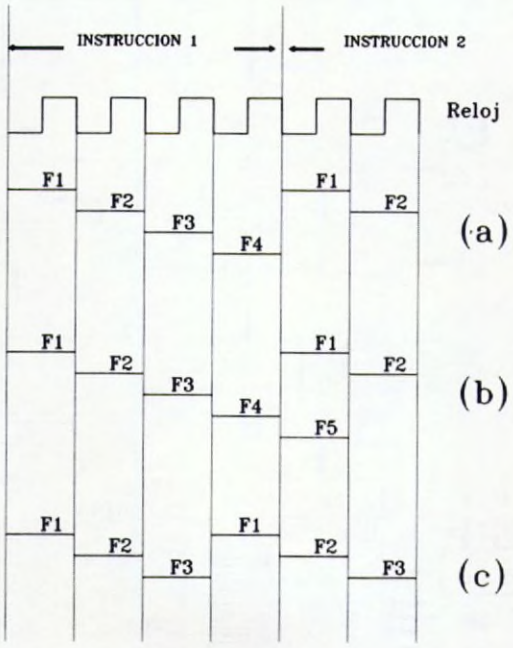


Figura 14.19. Cronogramas de secuenciamiento

14.6. **SISTEMA AMPLIADO**

El sistema digital SD-1 puede ser ampliado y mejorado conectando registros auxiliares en la unidad de proceso, conectando entre sí los buses para acelerar las operaciones, ampliando la posibilidad de seleccionar operaciones de la ALU, etc. El SD-2 que se representa en la figura 14.20 recoge todas estas mejoras, describiéndose en los cronogramas de las figuras 14.21 a 14.24 las operaciones de carga, suma, almacenamiento e incremento del contador de programa. Este sistema permite direccionamientos implícitos, relativos, paginados, etc., siendo su juego de instrucciones más amplio y potente que el SD-1.

Este circuito permite realizar mayor número de direccionamientos que el SD-1, abarcando la práctica totalidad de las posibilidades:

- DIRECCIONAMIENTO DIRECTO ABSOLUTO: La instrucción contiene la dirección real del operando. Si indica información sobre un registro, contiene la dirección del mismo.

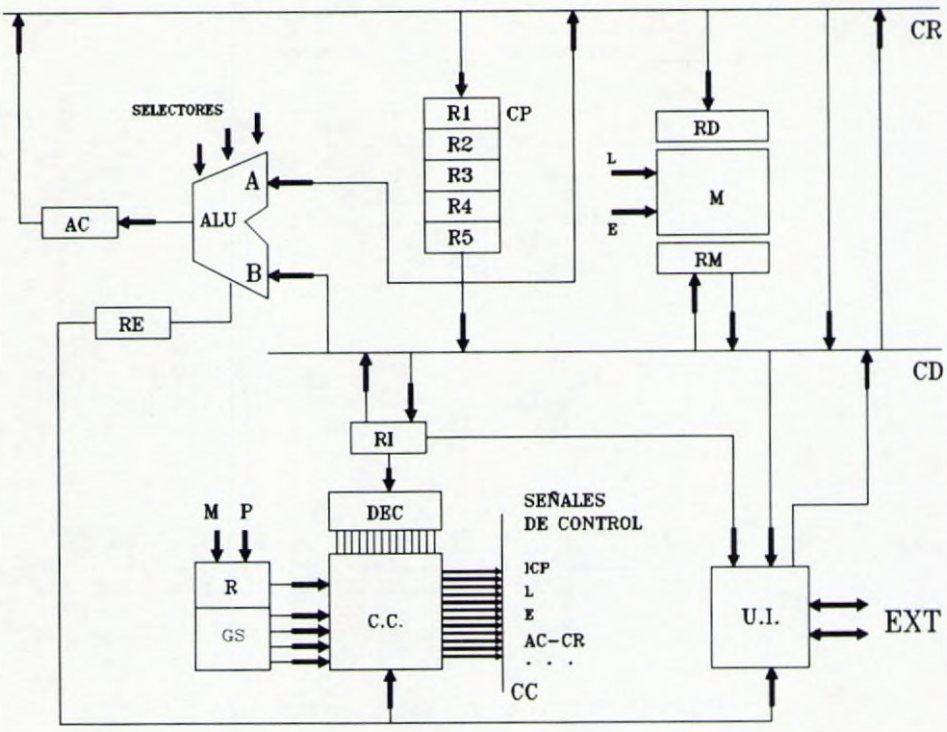


Figura 14.20. Sistema digital SD-2

- DIRECCIONAMIENTO INDIRECTO: La lectura de una posición de memoria proporciona la dirección del dato, no siendo necesario volver a pasar por RI para realizar la nueva lectura en M. Puede actuarse sobre M desde CP o uno de los registros y también desde RI.

- DIRECCIONAMIENTO INMEDIATO: La instrucción contiene el dato en forma de constante numérica.

- DIRECCIONAMIENTO DIRECTO RELATIVO O INDEXADO: En uno de los registros existe un puntero de referencia de la dirección en memoria del operando. A dicho valor se le suma un desplazamiento proporcionado normalmente por RI. Si se utiliza CP se denomina relativo a contador de programa, y si se destina un registro fijo para tal fin, se denomina relativo a registro base (si no varía el puntero) o registro índice (si varía el puntero).

- DIRECCIONAMIENTO PREINDEXADO: Se efectúa un direccionamiento relativo seguido de uno indirecto para obtener el operando.

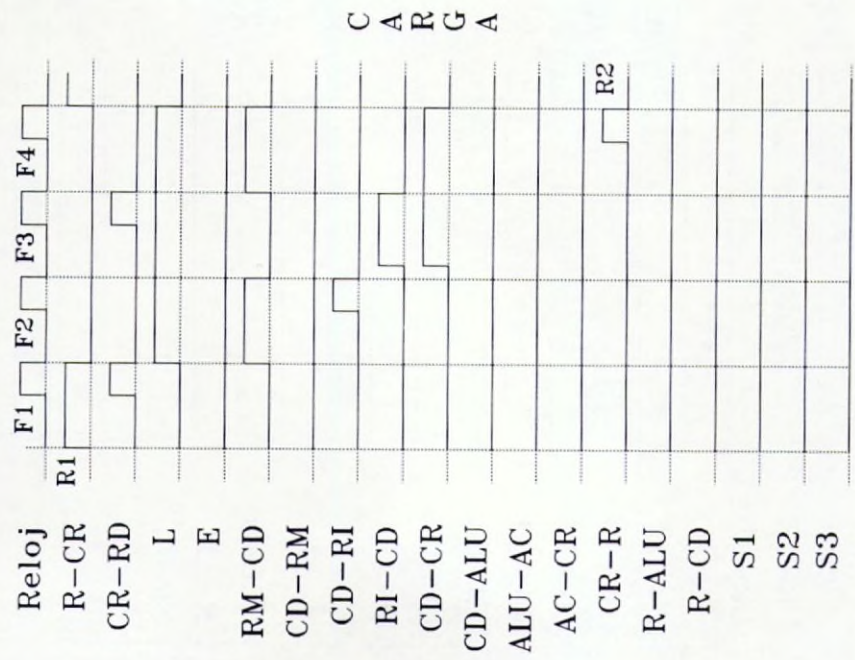


Figura 14.21. Cronograma de carga

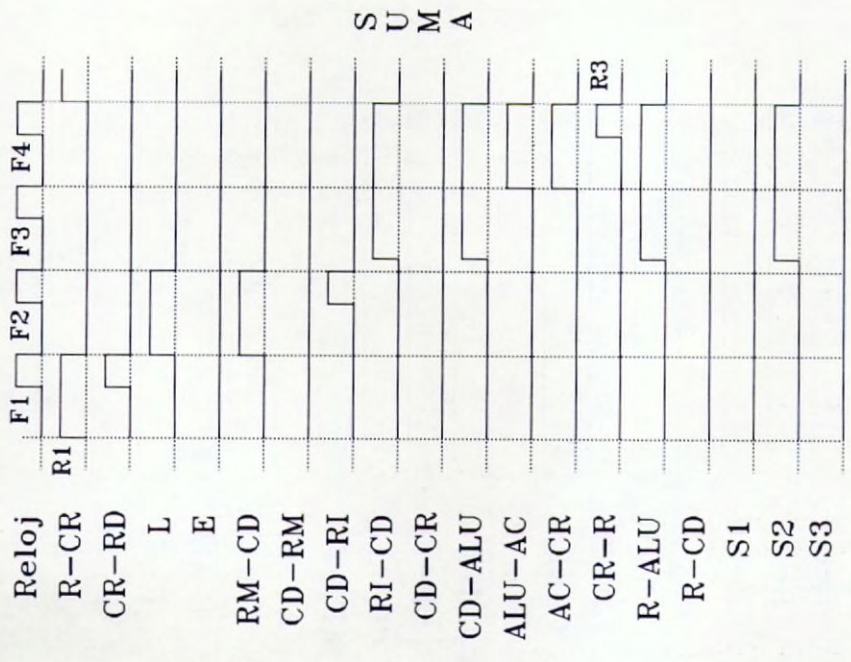


Figura 14.22. Cronograma de la suma

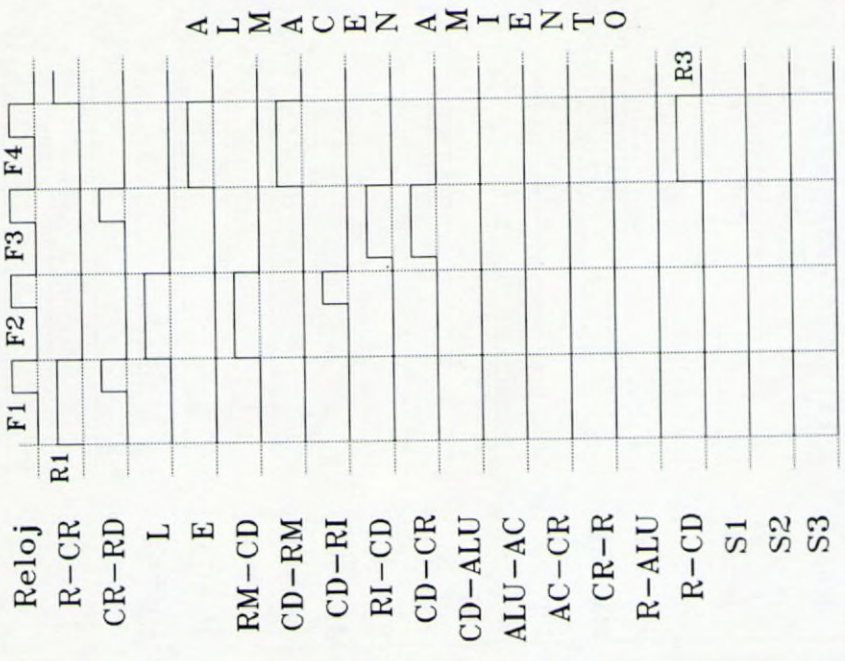


Figura 14.23. Cronograma de almacenamiento

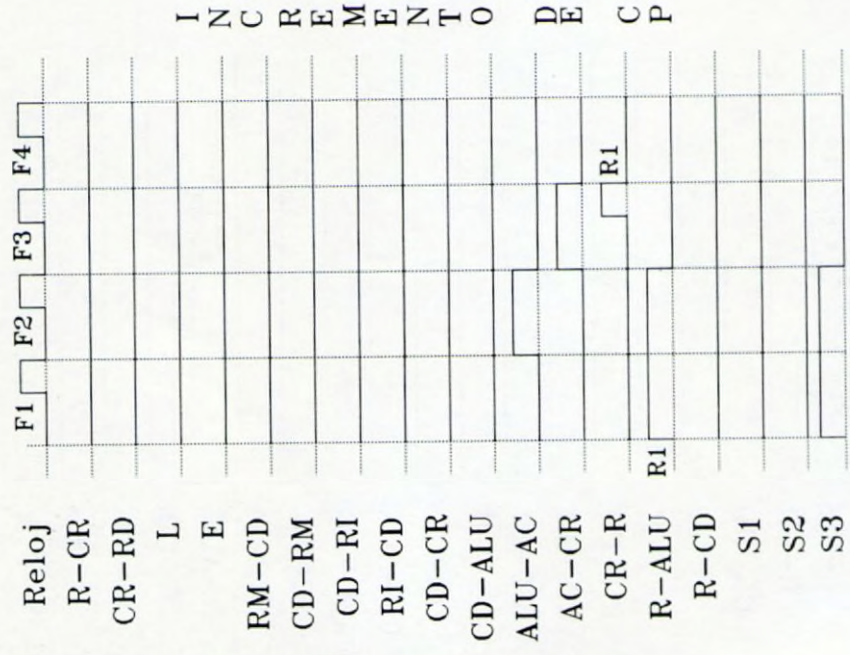


Figura 14.24. Cronograma de incremento de C.P.

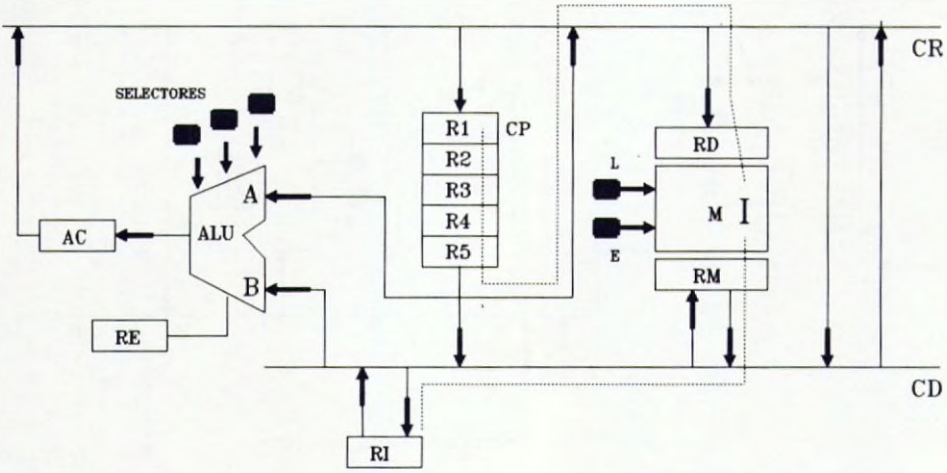


Figura 14.25. Direccionamiento directo absoluto - Primer ciclo

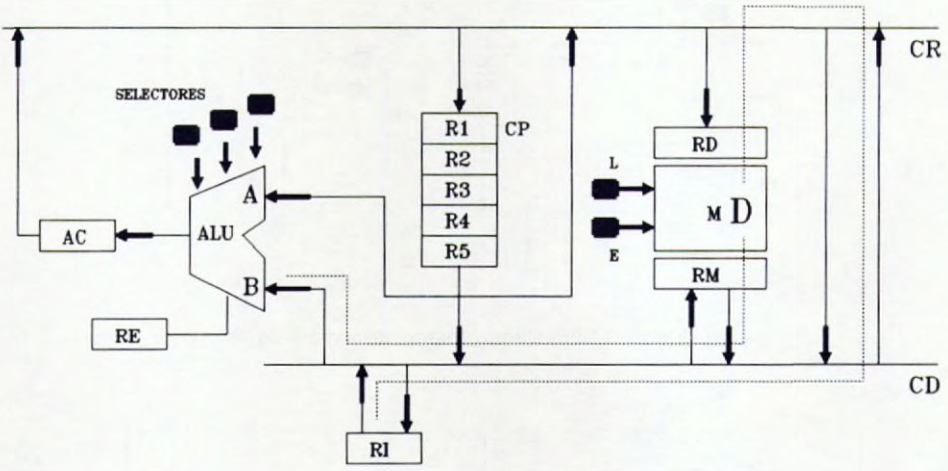


Figura 14.26. Direccionamiento directo absoluto - Segundo ciclo

- DIRECCIONAMIENTO POSTINDEXADO: Se efectúa una lectura en M para obtener un dirección parcial que sumar al puntero de indexación y así obtener la dirección definitiva.

- DIRECCIONAMIENTO IMPLICITO: No indica ninguna posición de memoria por lo que se direcciona el acumulador.

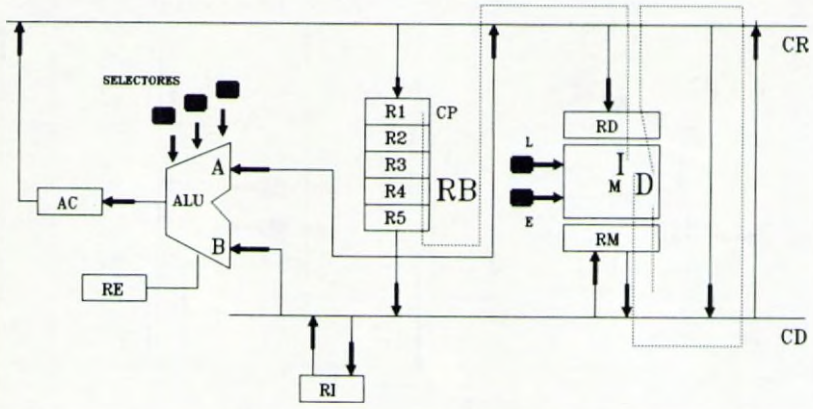


Figura 14.27. Direccionamiento indirecto

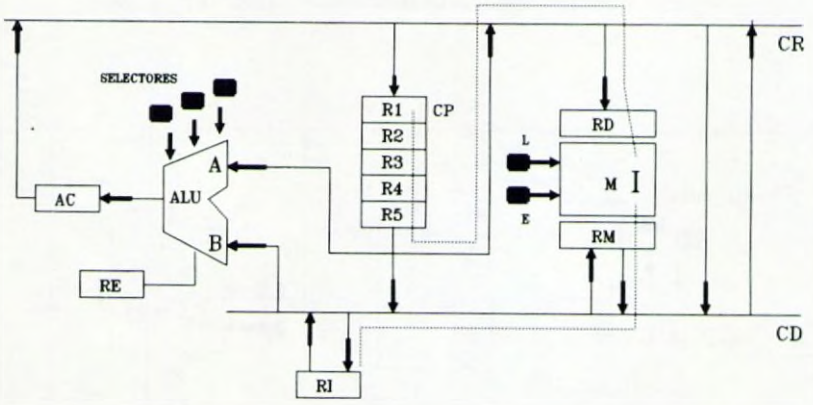


Figura 14.28. Direccionamiento inmediato

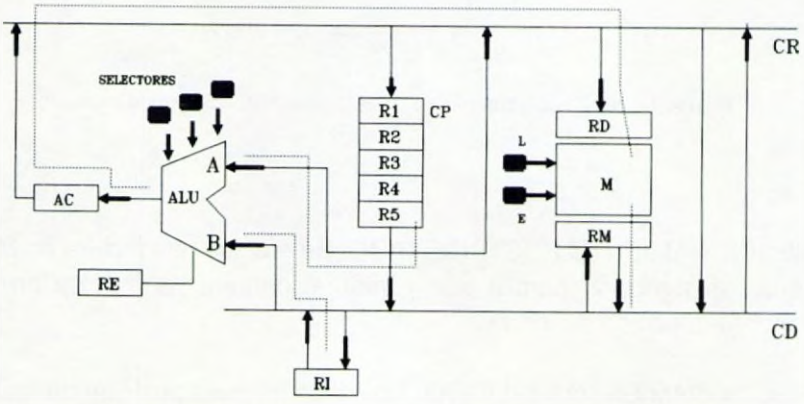


Figura 14.29. Direccionamiento indexado o relativo

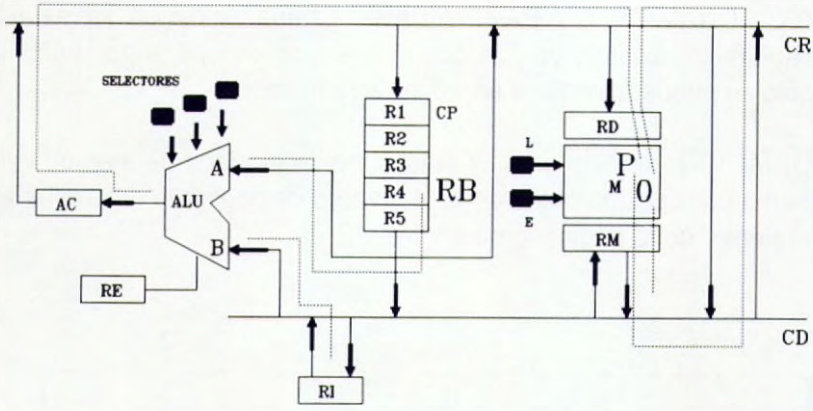


Figura 14.30. Direccionamiento preindexado

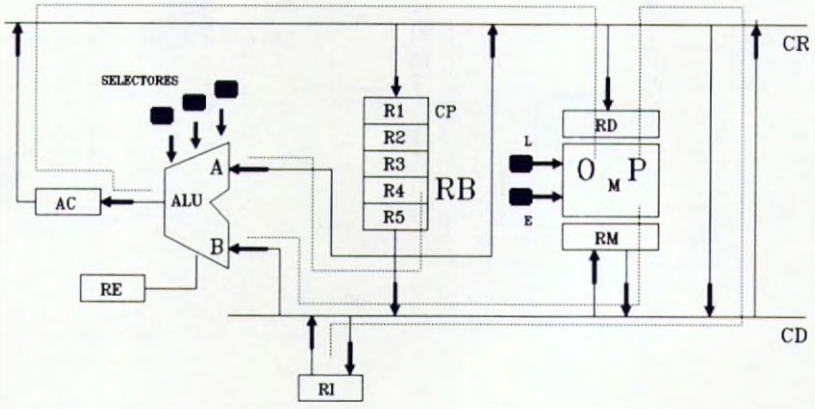


Figura 14.31. Direccionamiento postindexado

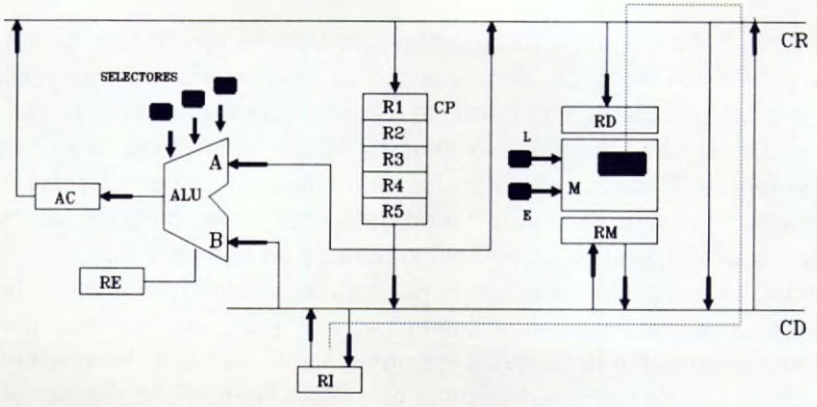


Figura 14.32. Concepto de página base (Memoria limitada)

- DIRECCIONAMIENTO A PAGINA BASE: Es una expresión que se le suele dar al directo absoluto ya que con la dirección de operando contenida en RI sólo se puede acceder a un trozo de la memoria.

- PAGINACION: Permite ampliar notablemente el mapa de memoria ya que se forma un registro doble donde el número de página lo determina RI y la línea dentro de la página lo determina CP.

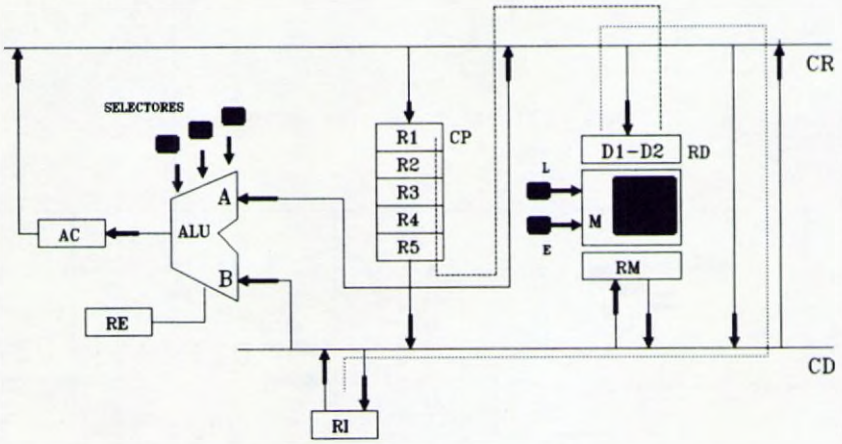


Figura 14.33. Paginación

14.7. **TRATAMIENTO DE INTERRUPCIONES. PRIORIDADES**

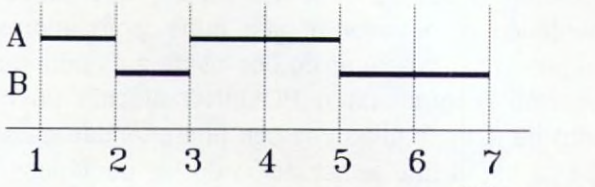
Durante el funcionamiento normal de un sistema procesador tienen lugar numerosas situaciones de simultaneidad de operaciones, en las cuáles se hace preciso establecer una prioridad. Si esto ocurre en el interior del sistema, también tendrá lugar en el exterior del mismo, sobre todo en su conexión con periféricos. Cuando son varios los elementos exteriores conectados a un sistema es necesario decidir el orden de funcionamiento de los mismos en caso de querer utilizarlos en un mismo período de tiempo.

Cada sistema microprocesado tiene unas posibilidades diferentes de establecer las prioridades de sus elementos, existiendo equipos que permiten tanto mecánicamente (Cableadas) como mediante software (Programadas) el establecimiento de dicho orden, otros que están semipreparados por el fabricante y sólo es necesario fijar determinados criterios de orden y otros que

imposibilitan al usuario fijar una ordenación determinada y han de aceptar la establecida por el fabricante.

Como el tratamiento de interrupciones y fijación de prioridades es un tema muy extenso, lo que interesa en proporción al sistema tratado y al punto de vista del análisis del mismo es comprender cómo funciona genéricamente un sistema como los descritos, teniendo en cuenta que puede disponer de varios dispositivos externos de comunicación con la unidad de proceso. Suponiendo que en este sistema se pueden fijar las prioridades según las necesidades que se tengan de uso del mismo, se pueden analizar por ejemplo 2 periféricos conectados A y B, asignándole al periférico A la prioridad 1 (máxima prioridad) y al periférico B la prioridad 2 (mínima prioridad).

Si el sistema está operando con el periférico A nunca interrumpirá la comunicación existente entre ambos hasta que finalice el trabajo que se está realizando. Si el sistema está operando con el periférico B, interrumpirá la comunicación existente entre ambos cuando el periférico A solicite comunicación, dado que se ha establecido mayor prioridad para A. Una vez terminada la comunicación entre el sistema y el periférico A, continuará el periférico B donde se había interrumpido. En la figura 14.34 se puede observar un gráfico de interrupciones entre un sistema central y dos periféricos A y B con la asignación de prioridades que se acaba de realizar. En este ejemplo se han determinado los instantes de tiempo en que se activa cada periférico y la duración de la ejecución.



Secuencia

t=1 seg A debe actuar durante 1 seg
t=2 seg B debe actuar durante 3 seg
t=3 seg A debe actuar durante 2 seg

Figura 14.34. Gráfico de prioridades

Esta especie de juego de conexiones ha de ser tratado con suma importancia. Se ha mencionado antes que estas interrupciones y prioridades que se han analizado entre el sistema y dos periféricos denominados A y B pueden ser también aplicables al interior de un sistema, entre elementos del

mismo, a diferentes programas o subprogramas residentes en la misma memoria del sistema, etc. Ello quiere decir que es necesario y fundamental resolver el posible problema que puede surgir tanto para la información que "espera" como para la información que ha de ser "retirada" momentáneamente para dejar su sitio a otra.

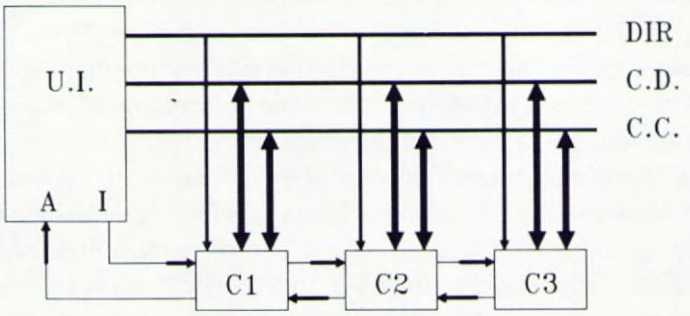


Figura 14.35. Prioridad cableada

La asignación de prioridades puede efectuarse mediante lógica cableada, programada o mixta. La asignación cableada o DAISY-CHAIN se efectúa mediante conexión a través de los controladores de un cable que determina las prioridades según el orden de instalación. La UCP a través de la unidad de intercambio envía una señal de interrupción que puede ser aceptada por cualquier periférico, bloqueando el paso a los controladores de inferior prioridad pero pudiendo ser interrumpido por los de más prioridad.

La asignación programada o POLLING efectúa un muestreo continuo (multiplexado) de los controladores con una prioridad determinada mediante software, de tal forma que la actuación de los periféricos será interrumpida cuando se cumpla lo establecido en la máscara de condición del programa. Suele actuar normalmente sobre una estructura multicalle de conexión de periféricos. La asignación mixta engloba ambas vertientes, reduciendo el trabajo de multiplexación y posibilitando un acceso externo mediante hardware a una rápida modificación de la prioridad de algunos periféricos.

Cuando una información que se está procesando bien sean datos o instrucciones de programa, ha de ser sustituida por otra de mayor prioridad, es necesario que la información inicial sea almacenada en la memoria del sistema para poder seguir procesándose a continuación. Cuando un programa llama a otro para que se realicen determinados cálculos y operaciones, es necesario saber a qué posición del programa ha de regresar una vez terminada la ejecución del nuevo programa al que se ha accedido. Para ello ha de

ser almacenada la línea de programa o la posición de memoria del mismo así como los parámetros y cálculos que se estaban utilizando en ese momento.

Cuando varios periféricos procesan simultáneamente con distintas prioridades, es necesario almacenar la información que luego por distintos procedimientos según los equipos, ha de ser distribuida entre los mismos. Con estos distintos ejemplos se pueden observar las variadas operaciones que el sistema ha de realizar para evitar la pérdida de información y especialmente la división y uso de determinadas zonas de la memoria de los equipos para almacenamiento temporal de la información que se está procesando. Por la importancia que estas zonas de memoria tienen, especialmente en cuanto a las posibles prestaciones de los equipos, no sólo se fabrican sistemas microprocesadores con varias zonas de memoria, sino que los elementos periféricos también disponen de memorias de capacidad relativamente pequeña comparado con la unidad central de los equipos, pero suficientes para permitir el uso simultáneo de los mismos y tratar de reducir al mínimo el número de interrupciones y los problemas que éstas podrían causar.

Se suelen denominar memorias tampón o buffers las memorias de los elementos periféricos, por ejemplo impresoras, cuya capacidad puede variar según el tipo de las mismas (impresoras de líneas, de páginas, gráficas, etc.) desde un valor muy normal de 8Kb en una impresora matricial de líneas hasta varios millones en impresoras gráficas, por ejemplo 2Mb en una láser. La posibilidad de disponer de estas memorias accesorias a la memoria del sistema central permite que la velocidad de respuesta del sistema y de cada uno de los elementos que lo componen tanto internos como externos pueda ser mayor. Esta velocidad se mejora dado que la unidad central puede funcionar en simultáneo con otros elementos, y no debe esperar a que un periférico, por ejemplo una impresora que se ha mencionado anteriormente, termine de imprimir una línea de caracteres para poder empezar a procesar los datos de la siguiente línea de impresión. En la figura 14.36 se puede apreciar el efecto de trabajar con y sin simultaneidad.

El acceso desde un periférico al interior del sistema es como puede apreciarse un aspecto muy complejo dentro del diseño de los sistemas digitales. La mayor parte de los comentarios realizados se basan en los ya clásicos sistemas de acceso directo a memoria por robo de ciclo y memoria multipuerta. El ROBO DE CICLO consiste en detener la ejecución de la UCP durante un ciclo para introducir datos. La petición externa de interrupción requiere la finalización del trabajo actualmente en ejecución.

La MEMORIA MULTIPUERTA consiste en disponer de varios registros juegos de registros de direcciones RD y auxiliares de memoria RM, uno para la UCP y el resto para los controladores de periféricos. Estos controladores se comunican al mismo tiempo con la UCP por lo que pueden detener su

ejecución para trabajar directamente con la memoria. Estos controladores poseen por tanto circuitos de control y secuenciamiento lo que les confiere la denominación de CANALES DE ACCESO DIRECTO A MEMORIA. En la figura 14.37 se hace un esquema de las conexiones de un canal y la UCP en una memoria multipuerta.

Uno de los elementos que normalmente requieren de un tratamiento especial de interrupciones, así como de un pequeño buffer de almacenamiento temporal de información, es el teclado de entrada de datos. Existen varias

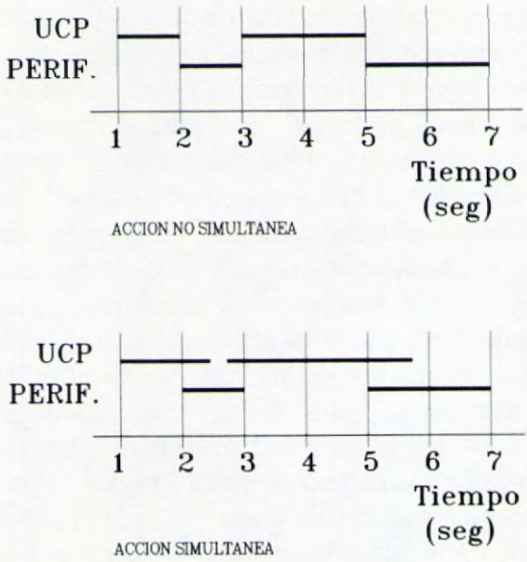


Figura 14.36. Simultaneidad

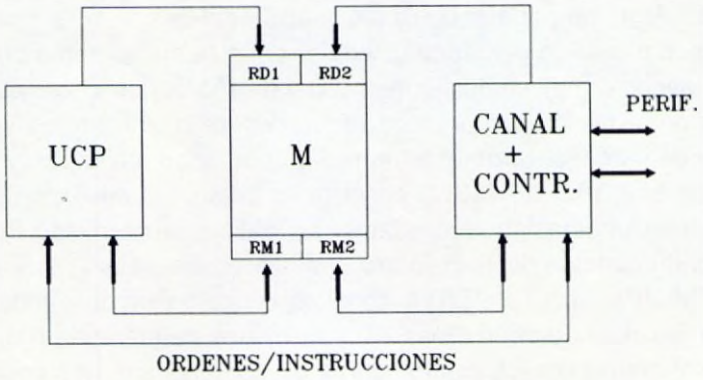


Figura 14.37. Memoria de doble puerta

vertientes de diseño de este tipo de equipos, pero básicamente casi todos los fabricantes se han decantado por utilizar el esquema básico que se indica en la figura 14.38. La UCP realiza un muestreo periódico del mismo a través del bloque denominado muestreador. Este elemento además convertirá a código ASCII o EDCDIC la señal pulsada y la transmitirá al interior del equipo procesador.

Los bloques B1 y B2 suelen ser registros, aunque B2 puede ser también un decodificador. El funcionamiento del mismo consiste en generar por parte de la UCP a través del muestreador una única señal activa que cíclicamente se irá almacenando en B2, produciendo en el esquema indicado como ejemplo los códigos 0001, 0010, 0100, 1000 y así repetidamente. Cuando se actúa sobre una tecla la señal activa "1" existente en B2 llegará a B1 (registro+triestados+diodos limitadores de tensión), obteniéndose de esta forma en el muestreador un código de 8 cifras binarias que en ASCII o EBCDIC equivale al caracter activado.

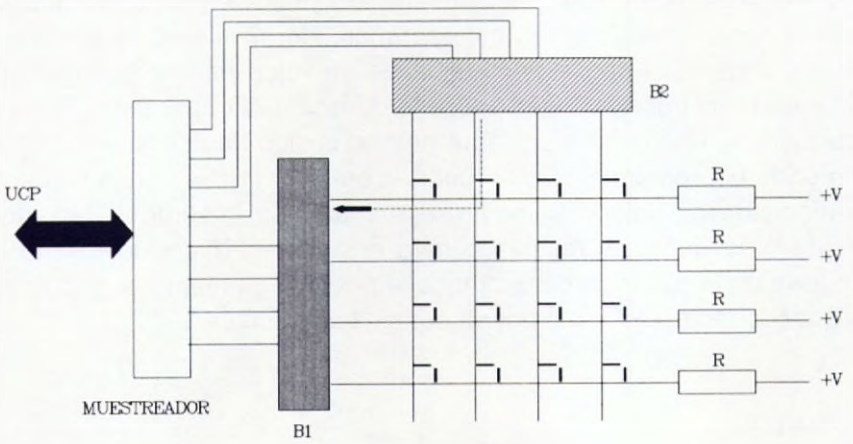


Figura 14.38. Teclado

14.8. **INSTRUCCIONES**

Las instrucciones que posea un ordenador determinarán las posibilidades de operación del mismo. Según la longitud de palabra que vaya a procesar el sistema, así serán las instrucciones y el número de ellas. Las longitudes de palabra más usuales son las de 8, 16 y 32 bits, especialmente en equipos pequeños y medianos. Estas longitudes de palabra han de ser suficientes para procesar los datos que se manejen bien en un sólo ciclo o en dos, dado

que un excesivo número de ciclos para obtener un dato haría el sistema excesivamente lento e inoperante. Teniendo en cuenta que a un sistema sencillo se le ha de dotar de un número mínimo de instrucciones, como las de almacenar, tomar de la memoria, saltar, sumar, complementar, parar, etc., (en inglés las conocidas LOAD, STORE, STOP, etc.), es necesario definir una referencia o código que identifique cada una de ellas. Lo primero por tanto que va a contener una de nuestras instrucciones es el código de la operación que se va a realizar. Según la longitud de palabra a emplear así será el número de bits que se podrán utilizar para la codificación de las distintas operaciones. La segunda parte de las instrucciones deberán contener la dirección del operando que se deba utilizar en la unidad de proceso.

Se puede definir una instrucción genérica como un conjunto de bits divididos en dos partes, la primera de ellas constituida por el código de la operación y la segunda por la dirección del operando a utilizar. Existen aplicaciones especiales denominadas operando inmediato y direccionamiento indirecto que modifican dicha constitución básica. El operando inmediato determina que la instrucción ya lleva el dato incluido, ocupando la zona de la dirección del operando. Normalmente se emplea con valores constantes y lógicamente limitado su valor máximo por el número de bits de la instrucción reservado a la dirección del operando. Si en una instrucción de 16 bits se reservan 4 para el código de operación y 12 para la dirección de operando, sólo se podrían emplear instrucciones de operando inmediato cuyo valor máximo no supere la potencia 10 de 2, dado que el bit número 12 debe ser reservado para el signo. El direccionamiento indirecto lleva en la dirección de operando la posición de memoria de otro dato, funcionando mediante la estructura denominada puntero.

14.9. **DATOS**

Normalmente durante la ejecución de un programa o simplemente durante el análisis de una parte del mismo se hace referencia a instrucciones de programa y a datos. Esta diferenciación de los elementos que intervienen en el programa viene establecida por la posibilidad de que determinadas partes del mismo puedan ser obtenidas mediante operaciones del sistema microprocesador. Se pueden denominar como instrucciones a los elementos básicos del programa sin los cuáles no sería posible su funcionamiento y que son necesariamente programados y definidos por el fabricante o el programador del sistema, dependiendo de los programas que se empleen.

Por tanto los datos serán los restantes elementos empleados por el programa, bien como información suministrada originalmente con las instruccio-

nes o bien obtenidas mediante operaciones aritméticas y lógicas a través de la unidad central de proceso del microprocesador. Estos datos que se manejan en el sistema pueden tener diversas estructuras, relacionadas todas ellas con su disposición y encadenamiento en la memoria, con lo cuál reciben diferentes denominaciones.

14.9.1. **Dato sencillo o simple**

La denominación genérica de dato se puede asignar al caso más sencillo de información, dado que el resto de los casos tendrá una denominación propia y diferente de las demás. El dato o "dato sencillo" es el elemento de información más simple, no relacionado con otro elemento ni descompuesto en factores y tiene una única dirección en la memoria donde poder ser localizado. Se puede considerar incluido en este grupo el caso del operando inmediato, en el cual un valor numérico (una constante de pequeño valor generalmente) está incluido en la zona de instrucción destinada a la dirección de operando.

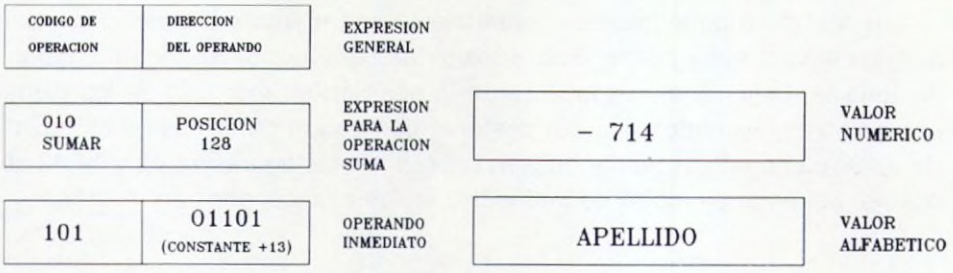


Figura 14.39. Formato de la instrucción básica

Figura 14.40. Dato simple

14.9.2. **Puntero**

Cuando algún dato ocupa una longitud de palabra superior a la de una posición de memoria se necesita almacenar dicha información en más de una posición de memoria. En este caso se fracciona dicha información en dos o más partes, las cuáles se grabarán en posiciones de memoria que pueden o no ser consecutivas. Estas direcciones estarán relacionadas entre sí mediante

lo que se denomina puntero. El puntero es por tanto el dato que en una parte del mismo, almacena la dirección o posición de memoria del siguiente dato.

En la figura 14.41 se representa un ejemplo de puntero en el cual un dato de longitud superior a la longitud de palabra de una posición de memoria ha sido fraccionado en dos partes. En la primera posición de memoria se ha incluido la dirección de la segunda posición de memoria.

14.9.3. **Estructuras matriciales**

Con bastante frecuencia el almacenamiento de datos de una memoria ha de realizarse en forma de tabla o de matriz, dado que existe una relación entre los distintos valores almacenados. En el caso de matrices, los datos se almacenan normalmente por filas correlativas, partiendo del primer elemento de la matriz. La localización de cualquier dato de dicha matriz se obtiene a partir de la posición que ocupa el primer elemento y de las sumas de los subíndices correspondientes. Las listas ordenadas y tablas pueden realizarse bien en forma matricial o en gran parte de los casos, debido a la entrada no consecutiva de datos para su almacenamiento en memoria mediante punteros.

Las tablas se corresponden normalmente con el almacenamiento en memoria de vectores, por lo que suelen también denominar como tablas vectoriales. Estos punteros localizarán la posición en memoria de los diversos datos almacenados, para la posterior confección de la tabla. Las tablas de datos con punteros se denominan índices. En las figuras 14.42 y 14.43 se pueden observar ejemplos de matrices y tablas almacenadas en memoria.

14.9.4. **Cadena de datos**

Aunque en algunos de los casos anteriores se permitía que algunos datos ocupasen posiciones no consecutivas de memoria, existía siempre alguna limitación. Esta limitación consiste en la ocupación consecutiva en memoria de la posición de algunos elementos de referencia, especialmente los punteros. En este caso de cadena de datos se prescinde por completo de cualquier limitación en cuanto a la ocupación consecutiva de la memoria, siendo por tanto de libre decisión por el programador o bien de ocupación por el sistema de la primera posición libre que encuentre. El caso más simple de cadena de datos sería el puntero, dado que enlaza dos elementos de la memoria que no tienen que estar en posiciones contiguas.

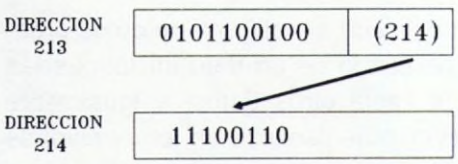


Figura 14.41. Puntero

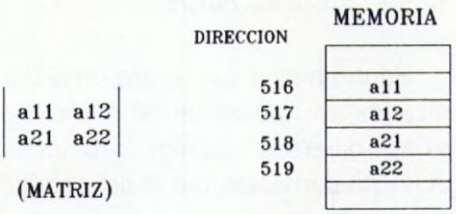


Figura 14.42. Matrices

NOMBRE	CALIFICACION	POSICION DE MEMORIA
JUAN	7,5	317
MIGUEL	4,3	435

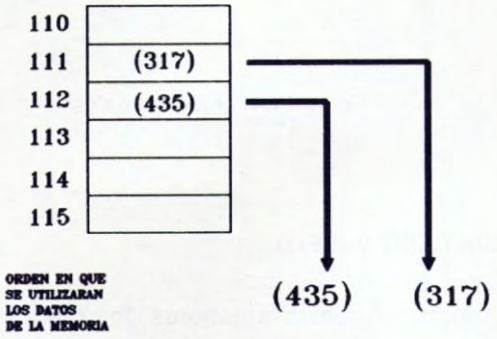


Figura 14.43. Tablas

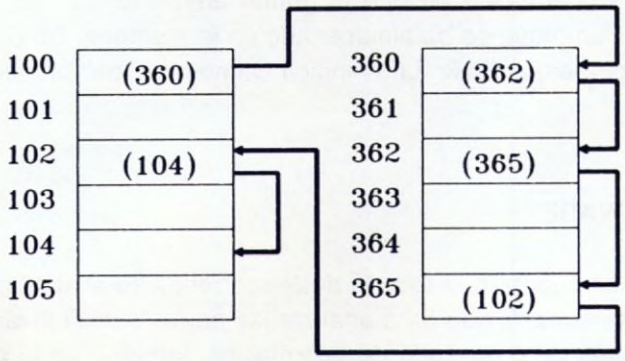


Figura 14.44. Cadena de datos

14.9.5. **Arbol de datos**

Se denomina estructura en árbol o ramificada al conjunto de datos organizados en niveles de tal manera que partiendo de un dato inicial pueden existir diversos caminos o bifurcaciones hacia otros datos, e igualmente ocurriría con estos datos del segundo nivel, obteniéndose un tercer nivel de datos y así consecutivamente. Un ejemplo de esta estructura ramificada se puede observar en la figura 14.45, donde se han construido cuatro niveles de información.

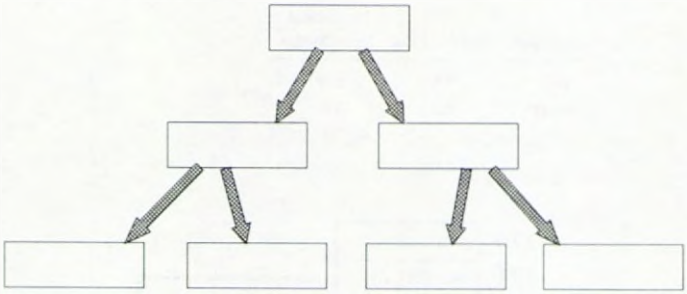


Figura 14.45. Arbol de datos

14.9.6. **Pila y Cola (LIFO y FIFO)**

Como ya se detalló en temas anteriores, los modos de almacenamiento FIFO y LIFO en las memorias constituyen la subdivisión de acceso serie o secuencial. La información va entrando o saliendo de la memoria en forma secuencial o consecutiva según las posiciones de la misma que han sido ocupadas. Se diferencian en que el primer elemento en salir puede ser el primero o el último que se ha almacenado en la memoria. FIFO indica primero en entrar primero en salir. LIFO indica último en entrar primero en salir.

14.10. **SOFTWARE**

Teniendo en cuenta que todo lo detallado respecto al sistema de ejemplo SD-1 ha de ser amplificado para analizar las partes constitutivas de un sistema real y actual de procesado de información, también en lo relativo a instrucciones y programas ocurre lo mismo. Igual que para el SD-1 se han

detallado algunas instrucciones y formatos que podían ser útiles para una primera toma de contacto con un sistema microprocesador, para un sistema real es necesario crear todo un conjunto de instrucciones básicas de funcionamiento del equipo. Estas instrucciones son normalmente estructuradas en lo que se denomina "Código de máquina".

Este código de máquina se puede hacer equivalente a la traducción del conjunto de instrucciones básicas del equipo al lenguaje de símbolos y códigos internos que la máquina o sistema microprocesador es capaz de entender. Siempre en sistema binario, es obtenido a través de la utilización de un sistema traductor, ya que estamos haciendo su equivalencia con una traducción de instrucciones. El método que se usa para traducir las instrucciones que el operador puede introducir desde un teclado al sistema microprocesador, es la utilización del lenguaje conocido como ensamblador.

Se denomina lenguaje ensamblador a una forma especial de programar los sistemas, similar a la que se puede emplear mediante lenguajes BASIC, FORTRAN, etc., que accede directamente a la base del sistema informático y lo configura según el posterior uso del mismo. Mediante la programación en ensamblador se puede no sólo configurar el conjunto de instrucciones, sino realizar un completo sistema operativo formado por un conjunto de programas y subprogramas que constituyen las posibilidades generales de empleo del sistema informático. Generalmente, los actuales equipos informáticos vie-

Figura 14.46. Esquema general de un programa ensamblador

nen preparados de fábrica para utilizar determinados sistemas operativos, según modelos y fabricantes. Sin embargo, todos ellos pueden ser reprogramados mediante lenguaje ensamblador, o basados en el conjunto de instrucciones que el fabricante ha incluido en el equipo, se pueden diseñar diferentes sistemas operativos especialmente para aplicaciones técnicas de bastante complejidad.

A partir del programa fuente se obtiene mediante sucesivas etapas la obtención del programa objeto ejecutable según el juego de instrucciones establecido por el fabricante. Normalmente se necesitan dos pasos de ensamblaje para su obtención aunque existen ensambladores de un paso que utilizan un trozo de memoria como elemento auxiliar para evitar un segundo ensamblaje.

Una vez en posesión del sistema operativo correspondiente, se podrán utilizar los programas adecuados a cada aplicación, bien los existentes en el mercado o los diseñados por el programador según el sistema operativo que haya realizado. En el mercado existen numerosos programas para su empleo en cada uno de los sistemas operativos, no pudiendo ser intercambiables entre sí y siendo necesario utilizar en cada programa la versión correspondiente al sistema operativo empleado por el equipo que se esté utilizando.

Como ejemplo se pueden mencionar los conocidos sistemas operativos MS-DOS, UNIX, XENIX, etc. El conjunto de programas que se pueden emplear en los sistemas informáticos es lo que se denomina como SOFTWARE, diferenciándose del HARDWARE que está constituido por el equipo físico.

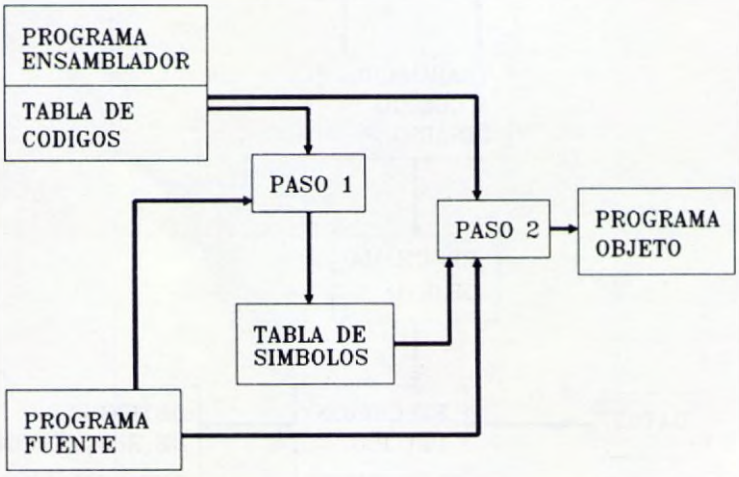


Figura 14.47. Ensamblador de dos pasos

Como ejemplo de programas utilizados por ellos, se pueden mencionar desde tratamiento de textos, hojas de cálculo, bases de datos, gráficos, diseño asistido, etc., de nombres bastante populares entre los usuarios de la informática, así como lenguajes de programación como el BASIC, COBOL, FORTRAN 77, ALGOL, RPG, C, PASCAL, etc. Esto origina muchas veces que a la hora de adquirir un equipo informático, bien sea para procesado de datos en general o para aplicación en control de procesos, sea necesario el análisis del sistema operativo que pueden emplear los diversos equipos y los programas existentes para el mismo. Debido a esta diversidad de opciones, cada vez son más los equipos que se suministran con la posibilidad de usar varios sistemas operativos, para evitar la limitación que el uso de uno sólo de ellos puede suponer. Incluso en esta vertiente de uso múltiple de sistemas operativos, se puede realizar simultáneamente el uso de varios sistemas operativos en un mismo equipo, mediante particiones de memoria para cada uno de los diferentes sistemas y trabajos en forma de multitarea, con las enormes posibilidades que ello conlleva. Este aumento de posibilidades de uso de los diferentes sistemas operativos y su software correspondiente, tiende a la creación de un sistema estandard para la mayor parte de los futuros equipos, al menos en la gama media y baja, constituidos por los miniordenadores y los ordenadores personales, con sus diferentes vertientes de conexión, como por ejemplo en la conexión mediante redes.

14.11. **APENDICE - MICROPROCESADORES COMERCIALES**

Se incluyen a continuación los diagramas de conocidos microprocesadores como el Z-80, 8085 y 8086 de Intel, así como sus coprocesadores y versiones más actualizadas (80286 y 80386). El Z-80 y el 8085 fueron pioneros en la fabricación de ordenadores personales y sistemas digitales de control. La figura 14.49 corresponde a la arquitectura interna del Z-80 y la figura 14.48 a la simbología de su circuito integrado. En la figura 14.50 se reproducen los terminales para el 8085. Este microprocesador presenta frente al Z-80 la autogeneración de la señal de reloj que sincroniza el sistema. En el Z-80 se necesita un oscilador exterior que genere dicha señal. Las posibilidades de tratamiento de interrupciones y de comunicación serie con otros sistemas van integradas igualmente en el 8085, microprocesador de estructura básica y posibilidades muy similares a las de los últimos microprocesadores aparecidos, el 80386 y 80486. Las salidas de los buses de datos y direcciones han de ser multiplexadas para permitir su uso compartido.

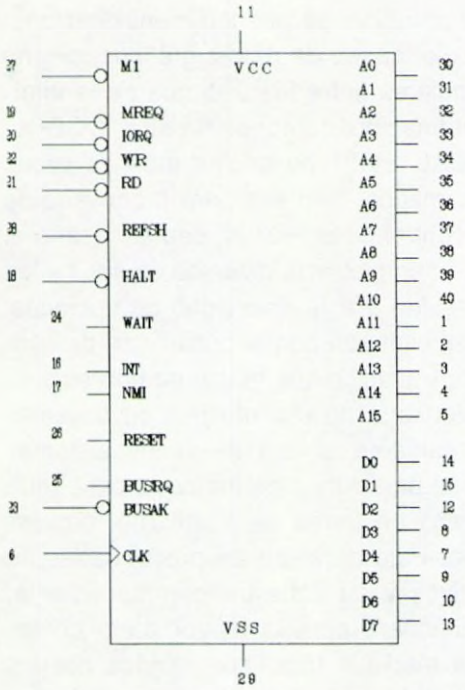


Figura 14.48. Configuración de terminales del Z-80

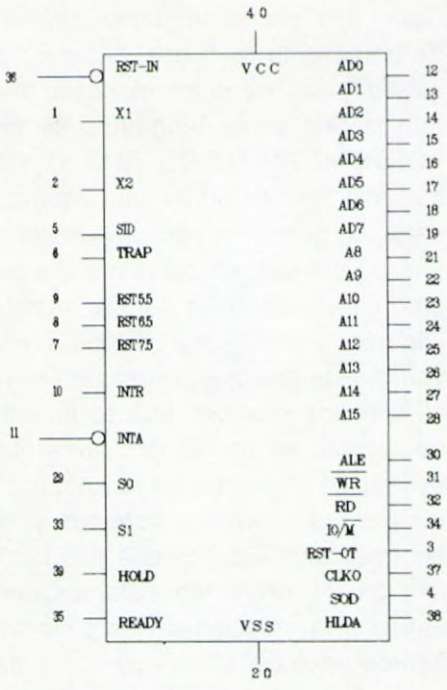


Figura 14.50. Configuración de terminales del 8085

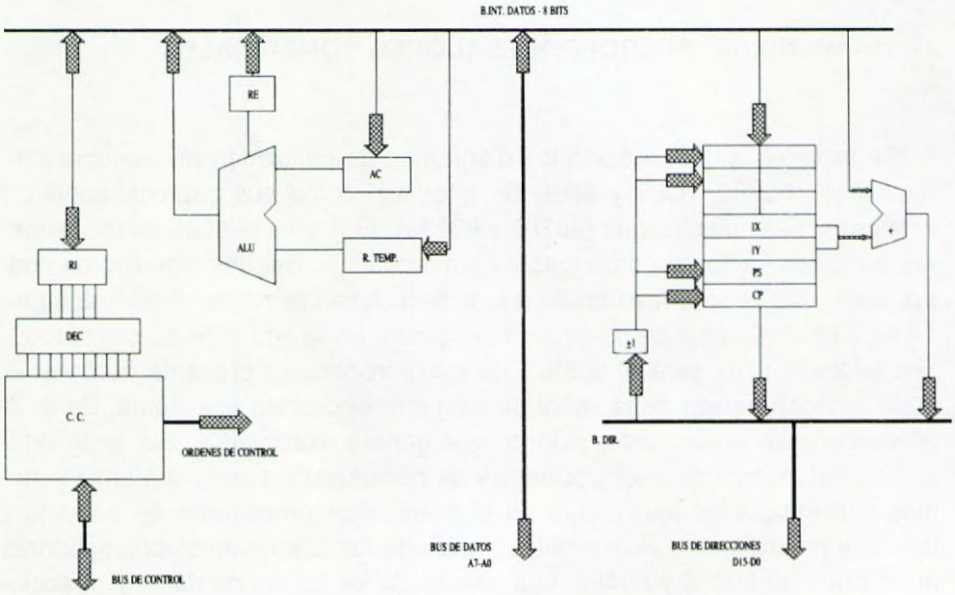


Figura 14.49. Arquitectura del Z-80

El 8085 por ejemplo trabaja a 3^{MHz}, direcciona 64K como máximo y trabaja con una alimentación de 5^{V}. Es un integrado de 40 terminales con bus de datos de 8 bits y bus de direcciones de 16 bits. Posee 6 registros de aplicaciones genéricas de 8 bits y registro de instrucciones con 5 banderas, además de 3 registros de 16 bits.

La separación de datos y direcciones es multiplexada externamente mediante la señal ALE (Address latch enable) según se aprecia en la figura

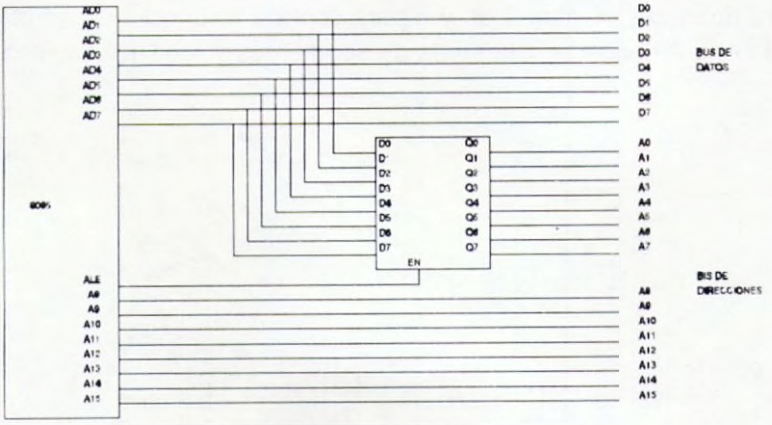


Figura 14.51. Separación de datos y direcciones

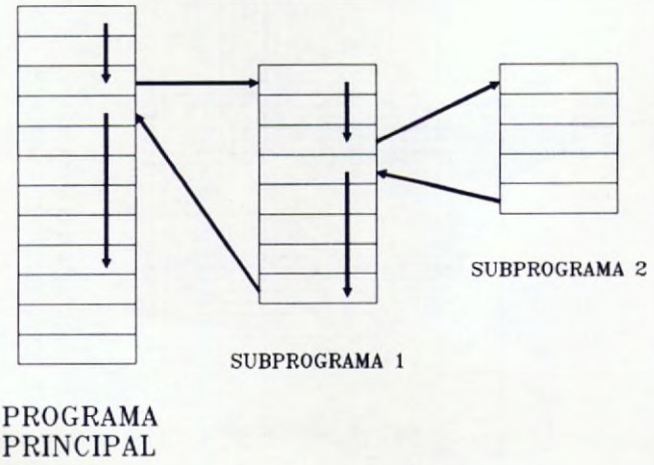


Figura 14.52. Llamada a subprograma y subrutinas

14.51. Posee acceso directo a memoria y mantiene en memoria principal una pila de sistema para almacenar las direcciones de retorno de las llamadas a subrutinas y de las interrupciones.

La figura 14.53 representa un montaje básico para la lectura de una memoria EPROM de 2KBytes y la lectura y escritura de una memoria RAM de 256 líneas de 8 bits.

El 8086 es un integrado de 40 terminales, con bus de datos de 16 bits y bus de direcciones de 16 bits. Igual que el 8085 necesita multiplexación exterior de datos y direcciones. Existen variantes con frecuencias entre 5 y 10^{MHz} con alimentación a 5^{V}. Los registros internos son de 16 bits, teniendo 14 para aplicaciones genéricas y 6 para cola de instrucciones. El registro de estado tiene 9 banderas y la suma de segmentos y registros índices permite

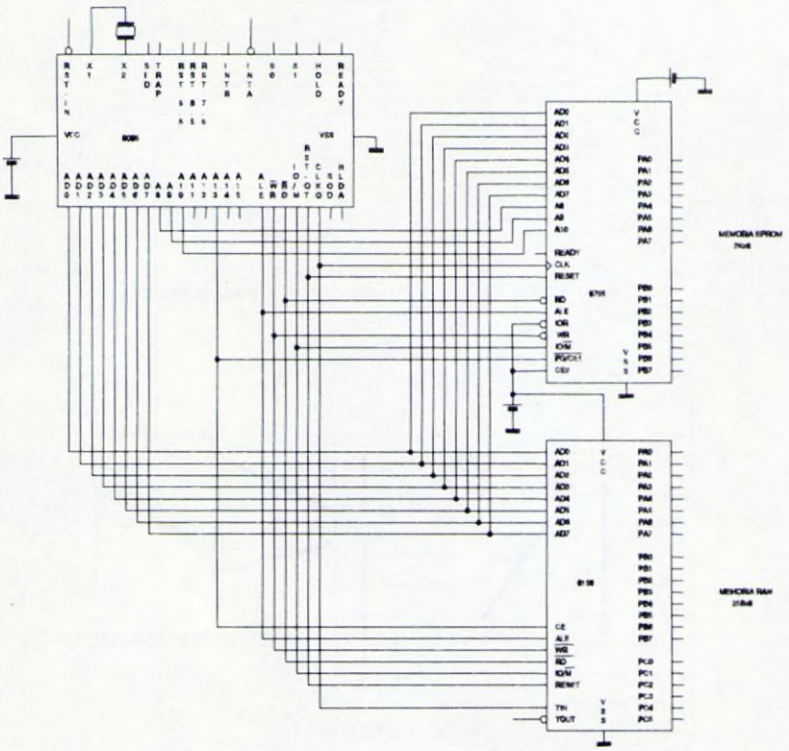


Figura 14.53. Esquema básico de conexiones de un 8085

obtener una dirección física de hasta 20 bits, lo que permite direccionar una memoria de 1MBytes. Tiene posibilidad de funcionar en modo mínimo en estructuras de un solo microprocesador y en modo máximo en montajes multiprocesadores. Las operaciones aritméticas de elevada complejidad las puede realizar a través de un coprocesador matemático (8087), gobernado mediante las señales de control del 8086. Una variante de este micro es el 8088, con un bus de datos de 8 bits.

Las versiones más modernas de este microprocesador son el 80286, circuito base de los ya clásicos ordenadores personales PC-AT y los 80386 y 80486, bloques básicos que operan con 32 bits, con velocidades por encima de 30^{MHz}, con mapas de memoria mínimos de varios megahertzios, memoria aceleradora cache y en el caso del 80486, con funciones coprocesadoras incorporadas.

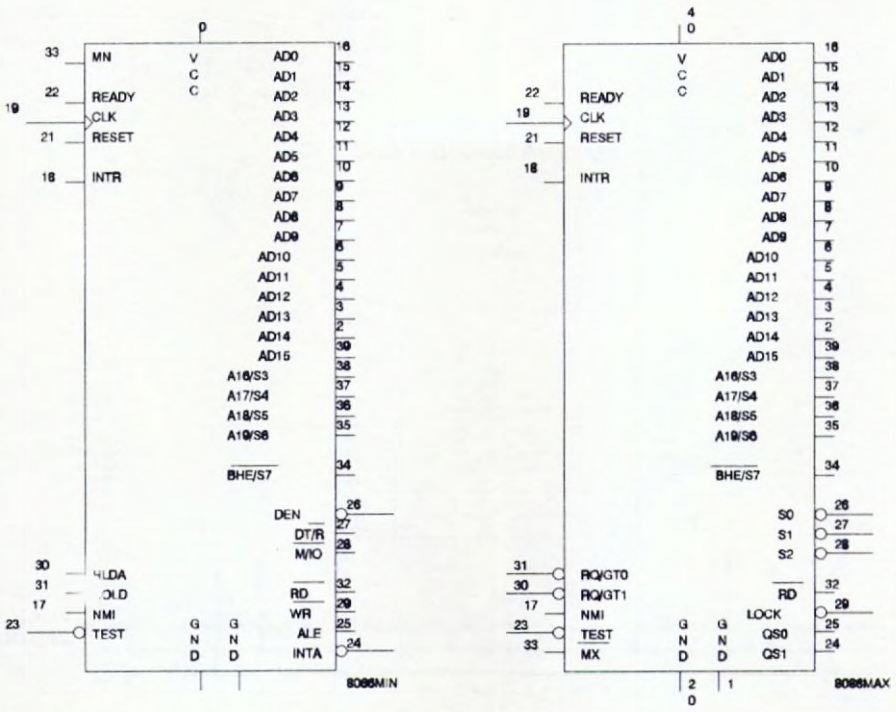


Figura 14.54. Configuraciones 8086

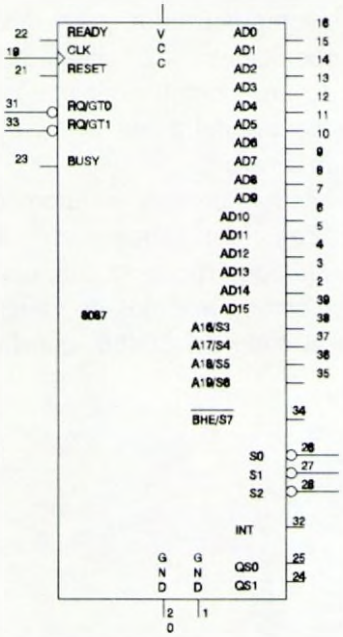


Figura 14.55. Coprocesador 8087

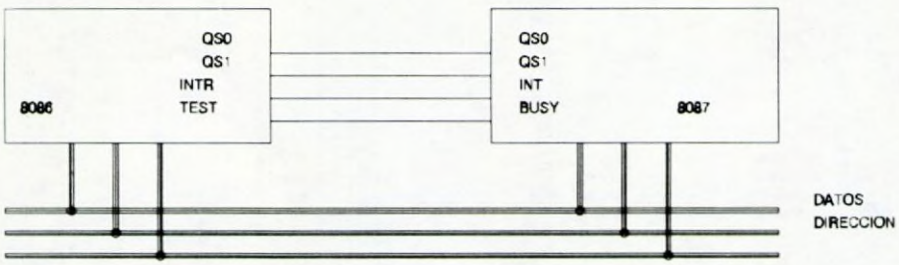


Figura 14.56. Conexión 8086-8087

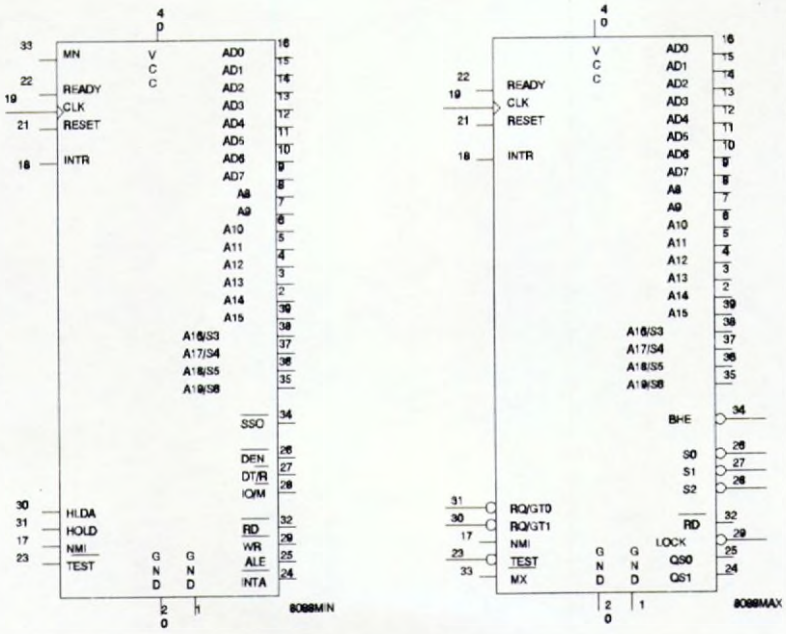


Figura 14.57. Configuraciones 8088

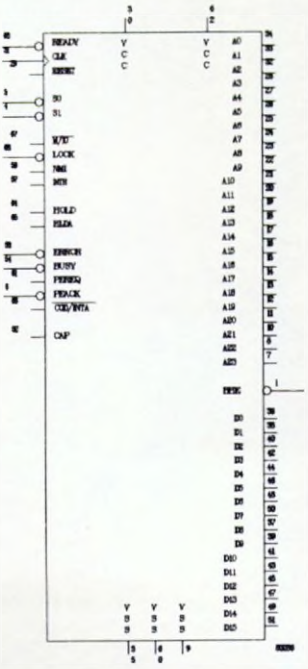


Figura 14.58. Configuraciones del 80286

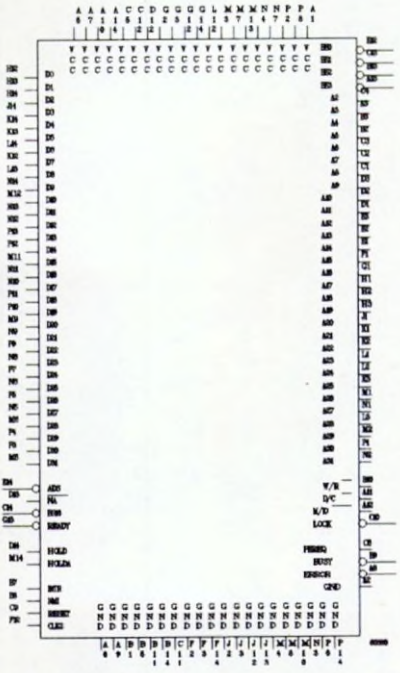


Figura 14.59. Configuraciones del 80386

APENDICE A
CUESTIONES Y PROBLEMAS

APENDICE A

2.1. Convertir en binario el número decimal 818.

2.2. Convertir en binario el número decimal 0,8.

2.3. Convertir en binario el número decimal 10,395.

2.4. Obtener el equivalente binario del número decimal 85,317 con 6 dígitos de precisión en la fracción.

2.5. Determinar el equivalente en sistema decimal del número binario 110011,10111.

2.6. Convertir al sistema decimal el número octal 1357.

2.7. Convertir al sistema octal el número binario 1111011.

2.8. Convertir al sistema decimal el número octal 1234,765.

2.9. Convertir al sistema octal el número decimal 25678.

2.10. Convertir al sistema octal el número decimal 625,87.

2.11. Convertir al sistema hexadecimal el número binario 01111001011.

2.12. Convertir el número decimal 142 al sistema de base 16.

2.13. Convertir el número decimal 0,66 a un número de base 8.

2.14. Convertir el número octal 756 a base 2.

2.15. Convertir el número hexadecimal BEBE a base dos.

2.16. Convertir el número octal 123456 a hexadecimal.

2.17. Convertir el número hexadecimal AAFF01,ED a octal.

2.18. Convertir el número binario 1101101101,1010001 a números en base octal y hexadecimal.

2.19. Obtener al número decimal equivalente al número 0010 1000 0100 en BCD natural.

2.20. Convertir el número 1100 1000 0011 perteneciente al código BCD exceso 3 a BCD natural, BCD Aiken y Binario natural.

2.21. Expresar el número decimal 23748 en BCD natural, BCD Aiken, BCD exceso 3, Binario natural y hexadecimal.

2.22. Calcular la frecuencia de una señal digital de período 1^{ns}.

2.23. Calcular el período de una señal digital de frecuencia 16^{MHz}.

2.24. Calcular el número de ciclos de reloj que se han generado en un circuito digital de frecuencia 8^{MHz} si ha funcionado durante 10^{seg}.

2.25. Si el área de un pulso es equivalente a 240^{Vs} y ha estado aplicado a un circuito digital de 5^{V}, calcular el instante en que dejó de funcionar el mismo.

2.26. Calcular dos escalones que generen el pulso del problema 2.25.

3.1. Convertir en decimal el número binario 1001100,101 representado en coma fija con 3 bits de parte fraccionaria y bit de signo.

3.2. Transformar a binario coma fija con bit de signo el número hexadecimal AB3,6F utilizando un registro de 16 bits.

3.3. Expresar el número decimal 217,85 en binario con signo y coma fija de 5 decimales, binario con coma fija de 10 decimales, octal, octal con signo, hexadecimal y hexadecimal con signo.

3.4. Expresar el número decimal -187,5 en binario con signo, binario con signo y coma fija de 4 decimales, binario con complemento a 1, binario con complemento a 2, octal y hexadecimal.

3.5. Representar en coma fija con bit de signo y 4 bits de parte fraccionaria en un registro de 16 bits el número decimal -2369,828125.

3.6. Representar en coma fija de 10 bits de parte entera y 1 de parte fraccionaria el número decimal 0,99999.

3.7. Representar en coma flotante para números enteros el número decimal 600,75 para un registro de 24 bits, 6 de los cuáles son del exponente.

3.8. Cuáles son los valores máximo y mínimo que se pueden almacenar en un registro de 32 bits con 8 para el exponente en formato de coma flotante para números enteros.

3.9. Representar en coma flotante para números enteros el número decimal 600,75 para un registro de 16 bits, 6 de los cuáles son del exponente.

3.10. Realizar el problema 3.7. para números fraccionarios no estandarizados, normalizados con bit implícito.

3.11. Realizar el problema 3.7. para números fraccionarios no estandarizados, normalizados sin utilizar bit implícito.

3.12. Realizar el problema 3.7. para números fraccionarios no estandarizados, no normalizado sin utilizar bit implícito.

3.13. Realizar el problema 3.7. para números estandarizados de simple precisión.

3.14. Realizar el problema 3.7. para números estandarizados de doble precisión.

3.15. Representar el número decimal -680,75 para un registro de 32 bits, con 6 de exponente en formato no estandarizado fraccionario.

3.16. Representar el número decimal -680,75 en un equipo VAX.

3.17. Utilizar el formato estándar para doble precisión y representar el número decimal -1.

3.18. Representar en formato directo de simple precisión el número decimal 146120000.

3.19. Añadir el bit de paridad impar al código BCD natural.

3.20. Añadir el bit de paridad par al código BCD exceso 3.

3.21. Partiendo de la tabla del código BCD natural, obtener los códigos de Hamming para corregir errores usando parida par y comprobar si existe error en la transmisión del número binario 1001100.

3.22. Transmitir por Hamming el número 24 y comprobar el error si se transmite el 25.

3.23. Transmitir por Hamming el número decimal 106 usando paridad par.

3.24. Se introduce en un equipo dotado de generador de códigos de Hamming una cifra de 5 bits. Si se transmite el número decimal 194, analizar si la transmisión ha sido correcta o no.

3.25. Al transmitir un código BCD Exceso-3 de 4 cifras, se ha cometido un error. Decir qué número decimal se quería transmitir, si la combinación obtenida incluyendo los códigos correctores de error es la 0110110. Se han usado códigos de control de paridad impar.

3.26. Calcular los códigos de Hamming para transmitir el número decimal 23, con paridad par.

3.27. Calcular los códigos de Hamming para transmitir el número decimal 1, con paridad impar.

3.28. Desarrollar y calcular las combinaciones necesarias para obtener los códigos de Hamming, cuando se desean transmitir números binarios de 6 dígitos.

3.29. Calcular los códigos de Hamming para transmitir el número decimal 3, con paridad par.

3.30. Transmitir consecutivamente las palabras AA, AB, AC, AD, AE y AF mediante códigos correctores de fila y columna con paridad par. Detectar y corregir un posible fallo en la palabra AE.

4.1. Simplificar el complemento de la ecuación:

$$A+AB+\overline{C}D$$

4.2. Encontrar el complemento de la ecuación:

$$A(\overline{B}+C(\overline{D}+\overline{E}))$$

4.3. Simplificar el complemento de la ecuación:

$$A(B+C)(D+\overline{A})$$

4.4. Simplificar la expresión siguiente:

$$A+B(B+\overline{C})$$

4.5. Demostrar que es cierta la siguiente igualdad:

$$\overline{A}B(\overline{D}+D\overline{C})+(A+D\overline{A}C)B=B$$

4.6. Hallar mediante la tabla de verdad, que es correcta la expresión:

$$(A+B)(\overline{A}+C)=AC+\overline{A}B$$

4.7. Obtener la salida de una puerta NAND si sus entradas son las salidas de dos puertas NOR.

4.8. Obtener la tabla de verdad de la función Y- EXCLUSIVA de tres variables a partir de la de dos variables.

4.9. Obtener la salida de una puerta NOR si sus entradas están conectadas ambas a la salida de una puerta AND.

4.10. Obtener la salida al conectar el interruptor, según el esquema de la figura A.1.

4.11. Obtener la tabla de verdad de la suma de tres variables lógicas.

4.12. Obtener la tabla de verdad del producto de tres variables lógicas.

4.13. Obtener la salida al conectar los interruptores I1, I2 según el esquema de la figura A.2.

4.14. Obtener el cronograma de salida del circuito de la figura A.1. al aplicarle una vez cerrado el interruptor la señal digital de la figura A.3.

4.15. Obtener el cronograma de salida del circuito de la figura A.2. al aplicarle una vez cerrados los interruptores la señal digital de la figura A.3 en ambas entradas.

4.16. Aplicar a las 3 entradas del circuito de la figura 4.21 la señal digital de la figura A.3 y obtener su salida.

4.17. Repetir el problema 4.16 con la figura 4.23.

4.18. Aplicar las entradas A, B, C del cronograma 4.24 al circuito de la figura 4.25 y obtener su salida.

4.19. Reducir el circuito de puertas lógicas de la figura A.4.

4.20. Aplicar las entradas A=1, B=0, C=1, al circuito de la figura A.4 y obtener su salida.

4.21. Aplicar las entradas A=1, B=1, C=0, al circuito de la figura 4.29 y obtener su salida.

4.22. Aplicar las entradas A=1, B=0, C=1, D=0 al circuito de la figura 4.30 y obtener su salida.

4.23. Aplicar las entradas A=1, B=0, C=1, D=0 al circuito de la figura 5.11 y obtener su salida.

4.24. Aplicar el cronograma de la figura A.5. al circuito de la figura A.4. y obtener gráficamente la salida.

4.25. Aplicar el cronograma de la figura A.5. al circuito obtenido en el problema 4.19.

5.1. Simplificar según los teoremas del álgebra de Boole, la función que se indica a continuación. Realizar el diagrama de puertas lógicas resultante de la simplificación.

$$F(A,B,C,D) = M_1 M_2 M_3 M_4$$

5.2. Simplificar según los teoremas del álgebra de Boole, la función que se indica a continuación. Realizar el diagrama de puertas lógicas resultante de la simplificación e indicar la salida en caso de que A=1, B=1, C=0 y D=1.

$$F(A,B,C,D)=ABC+\overline{A\bar{B}(A+\bar{C})+(BD+\bar{C})+\bar{B}}$$

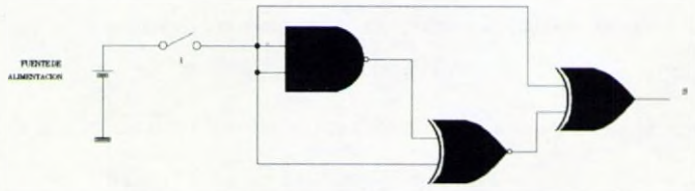


Figura A.1

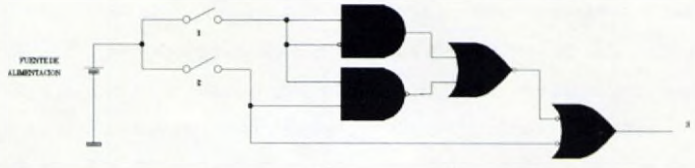


Figura A.2

5.3. Siendo F la función que se indica a continuación, hallar las expresiones canónicas de suma y producto y las numéricas.

$$F(A,B,C)=AB+AB\overline{C}+\overline{A}\overline{B}+\overline{A}BC$$

5.4. Representar la tabla de verdad del problema anterior.

5.5. Dada la función F siguiente, obtener las expresiones canónicas y la tabla de verdad.

$$F(A,B,C)=\overline{(A+B)}+\overline{A}BC+\overline{A(B+C)}$$

5.6. Dada la función F que se indica a continuación, obtener la expresión canónica de producto de sumas, las expresiones canónicas algebraicas y la tabla de verdad.

$$F(A,B,C,D)=m_0+m_1+m_3+m_5+m_7+m_9+m_{10}+m_{14}$$

5.7. Dada la siguiente tabla de verdad, obtener las expresiones canónicas:

A	B	C	D	F
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

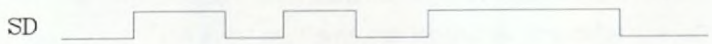


Figura A.3

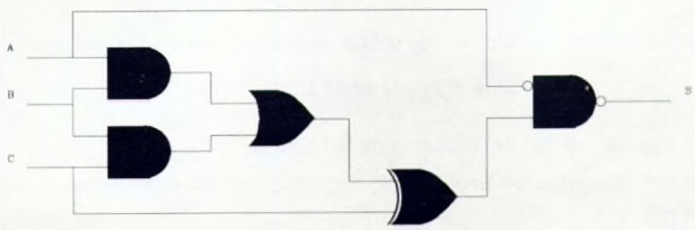


Figura A.4

5.8. Expresar la función F que se indica a continuación como suma de productos y producto de sumas.

$$F(A,B,C,D)=AB+BC\bar{D}+\bar{A}C\bar{D}$$

5.9. Dada la función F siguiente, expresarla como forma canónica de producto de sumas.

$$F(A,B,C,D)=(\bar{A}+BC)(B+\bar{C}D)$$

5.10. Simplificar por el método de Karnaugh la función siguiente:

$$F(A,B,C)=m_0+m_1+m_5$$

5.11. Hallar la tabla de verdad de la siguiente función:

$$F(A,B,C,D)=M_1M_2M_5M_6M_{11}$$

5.12. Obtener la función del problema 5.7.

5.13. Simplificar por el método de Karnaugh la función siguiente:

$$F(A,B,C,D,E)=m_0+m_2+m_7+m_8+m_{11}+m_{12}+m_{16}+m_{18}$$

5.14. Comprobar los resultados de la simplificación por el método de Karnaugh, utilizando las dos expresiones canónicas en la función:

$$F(A,B,C,D)=M_1M_2M_3M_5M_7M_{10}M_{11}$$

5.15. Simplificar por el método de Quine McCluskey la función del problema 5.14.

5.16. Simplificar por el método de Quine McCluskey la función del problema 5.13.

5.17. Representar con puertas NAND la función del problema 5.15.

5.18. Representar con puertas NAND la función del problema 5.1.

5.19. Representar con puertas NOR la solución del problema 5.15.

5.20. Representar con puertas NOR la solución del problema 5.13.

5.21. Representar con puertas NOR la solución del problema 5.14.

5.22. Representar con puertas NAND la solución del problema 5.14.

5.23. Obtener la tabla de verdad del esquema de la figura A.6.

5.24. Simplificar mediante el álgebra de Boole la siguiente expresión y representarla con puertas lógicas:

$$F(A,B,C,D)=(m_0+M_1+m_3+m_5+M_9+m_{11})(M_1M_7)$$

5.25. Simplificar mediante Quine McCluskey utilizando producto de sumas, la función:

$$F(A,B,C,D)=M_0M_1M_9M_{11}M_{15}$$

5.26. Obtener la tabla de verdad del esquema de la figura A.7.

5.27. Simplificar por el álgebra de Boole, la siguiente expresión y representar el circuito con puertas NAND:

$$F(A,B,C,D)=M_1M_5M_9M_{13}$$

5.28. Simplificar por el álgebra de Boole la siguiente expresión y representar el circuito con puertas NOR:

$$F(A,B,C,D,E)=m_0+m_1+m_4+m_7+m_{14}+m_{16}+m_{18}+m_{20}$$

5.29. Simplificar mediante Karnaugh utilizando productos de sumas, la función:

$$F(A,B,C,D)=m_1+m_2+m_5+m_8+m_9+m_{10}+m_{11}+m_{13}+m_{15}$$

5.30. Obtener la función que representa al esquema de la figura A.8.

5.31. Sea la función F que se indica a continuación. Obtener la tabla de verdad correspondiente, la expresión de F como suma de productos y simplificar por Quine McCluskey usando producto de sumas.

$$F(A,B,C,D)=M_0M_1M_5M_7M_8M_9$$

5.32. Sea la función F que se indica a continuación. Obtener la tabla de verdad correspondiente, la expresión de F como suma de productos y simplificar por Karnaugh usando producto de sumas.

$$F(A,B,C,D)=M_1M_2M_3M_4$$

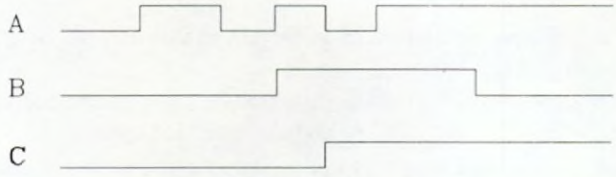


Figura A.5

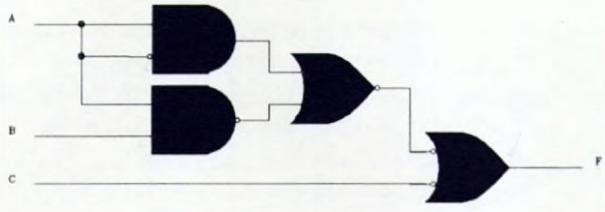


Figura A.6

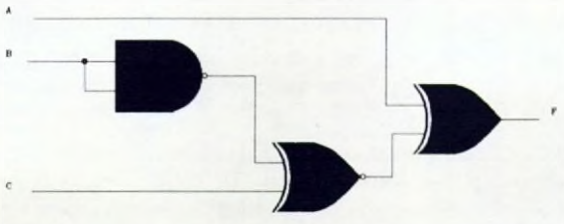


Figura A.7

5.33. Dada la función F siguiente, obtener la tabla de verdad, simplificar la función mediante Quine McCluskey y representar la función resultante mediante puertas NAND de 2 entradas.

$$F(A,B,C,D)=A\bar{B}\bar{C}+\overline{(A+\bar{B})+(\bar{D}+C)}+\bar{B}\bar{C}D+\bar{A}\bar{B}\bar{C}\bar{D}+ABC$$

5.34. Realizar el problema 5.29 empleando simplificación por Karnaugh de suma de productos.

5.35. Simplificar la función del problema 5.32 empleando Karnaugh producto de sumas.

5.36. Simplificar por Karnaugh la función de 5 variables:

$$F=m_0+m_3+m_5+m_6+m_{12}+m_{15}+m_{19}+m_{30}+m_{31}$$

5.37. Simplificar por Karnaugh la función de 5 variables:

$$F=M_0M_2M_3M_6M_7M_8M_{12}M_{20}M_{24}M_{28}$$

5.38. Simplificar por Karnaugh la función de 6 variables siguiente:

$$F=m_0+m_6+m_{12}+m_{20}+m_{24}+m_{25}+m_{28}+m_{32}+m_{36}+m_{50}$$

6.1. Representar gráficamente las señales de entrada y salida de una puerta NO de lógica TTL, 5V, $t_{PHL}=28^{ns}$ y $t_{PLH}=16^{ns}$.

6.2. Representar gráficamente las señales de entrada y salida de una puerta NOR de lógica TTL, 5V, $t_{PHL}=18^{ns}$ y $t_{PLH}=12^{ns}$.

6.3. Representar gráficamente las señales de entrada y salida de una puerta NAND de lógica TTL, 5V, $t_{PHL}=16^{ns}$ y $t_{PLH}=15^{ns}$.

6.4. Representar gráficamente las señales de entrada y salida de una puerta NOR de lógica TTL, 5V, $t_{PHL}=18^{ns}$ y $t_{PLH}=12^{ns}$, con una entrada conectada a masa.

6.5. Representar gráficamente las señales de entrada y salida de una puerta NOR de lógica TTL, 5V, $t_{PHL}=18^{ns}$ y $t_{PLH}=12^{ns}$, con una entrada conectada a un "1".

6.6. Representar gráficamente las señales de entrada y salida de una puerta NAND de lógica TTL, 5V, $t_{PHL}=16^{ns}$ y $t_{PLH}=15^{ns}$, con una entrada conectada a masa.

6.7. Representar gráficamente las señales de entrada y salida de una puerta NAND de lógica TTL, 5V, $t_{PHL}=16^{ns}$ y $t_{PLH}=15^{ns}$, con una entrada conectada a un "1".

6.8. Representar gráficamente las señales de entrada y salida de una puerta XOR de lógica TTL, 5V, $t_{PHL}=18^{ns}$ y $t_{PLH}=12^{ns}$, con una entrada conectada a masa.

6.9. Representar gráficamente las señales de entrada y salida de una puerta XOR de lógica TTL, 5V, $t_{PHL}=18^{ns}$ y $t_{PLH}=12^{ns}$, con una entrada conectada a un "1".

6.10. Representar gráficamente las señales de entrada y salida de una puerta AND de lógica TTL, 5V, $t_{PHL}=20^{ns}$ y $t_{PLH}=17^{ns}$, con una entrada conectada a masa.

6.11. Representar gráficamente las señales de entrada y salida de una puerta AND de lógica TTL, 5V, $t_{PHL}=20^{ns}$ y $t_{PLH}=17^{ns}$, con una entrada conectada a un "1".

6.12. Representar gráficamente las señales de entrada y salida de una puerta NAND con una entrada conectada a masa con frecuencia 4^{MHz}, tiempo de subida 7^{ns}, tiempo de bajada 10^{ns}, $t_{PLH}=22^{ns}$, $t_{PHL}=25^{ns}$.

6.13. Representar gráficamente las señales de entrada y salida del circuito de puertas lógicas formado por una puerta AND de lógica TTL, 5V, $t_{PHL}=20^{ns}$ y $t_{PLH}=17^{ns}$, con una entrada conectada a un "1", cuya salida se conecta a una entrada de una puerta NAND de lógica TTL, 5V, $t_{PHL}=18^{ns}$ y $t_{PLH}=19^{ns}$, estando la otra entrada conectada a un "1".

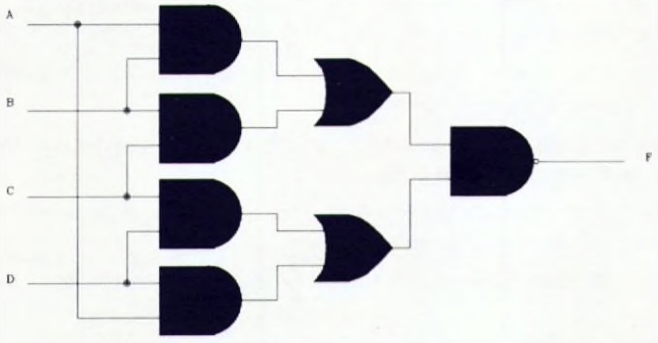


Figura A.8

7.1. Construir con demultiplexores de 2 canales un circuito donde sus salidas sean las entradas de un multiplexor de 8 canales. Indicar los valores de las entradas de selección que permitan que un "1" en la entrada del demultiplexor aparezca siempre en la salida del multiplexor.

7.2. Construir el esquema correspondiente a un circuito con 32 salidas decodificadas, usando decodificadores BCD a decimal. Indicar los valores de las señales de entrada para activar la salida vigesimosexta.

7.3. Generar con decodificadores la función del problema 5.8.

7.4. Generar con decodificadores la función del problema 5.10.

7.5. Generar con decodificadores la función del problema 5.14.

7.6. Generar con decodificadores BCD/Decimal la función del problema 5.29.

7.7. Generar con decodificadores BCD/Decimal la función del problema 5.13.

7.8. En el esquema de la figura A.9 se aplica la señal de entrada A=1. Obtener las señales activas a las salidas de los decodificadores, los valores de las señales de salida de los multiplexores y el rótulo luminoso que aparecerá en los displays 7 segmentos.

7.9. Representar con multiplexores la función utilizada en el problema 5.2.

7.10. Calcular las ecuaciones de diseño de un comparador de 3 bits.

7.11. Representar con multiplexores la función de 4 variables:

$$F=M_0M_1M_3M_5M_6M_7M_8M_{14}$$

7.12. Generar con multiplexores de 2 canales la función del problema 5.8.

7.13. Generar con multiplexores de 4 canales la función del problema 5.10.

7.14. Generar con multiplexores de 2 canales la función del problema 5.14.

7.15. Generar con multiplexores la función del problema 5.29.

7.16. Generar con decodificadores BCD/Decimal la función del problema 5.13.

7.17. Obtener el cronograma de salida del decodificador 3x8 aplicando en sus entradas las señales digitales de la figura A.10.

7.18. Obtener el cronograma de salida del codificador 8x3 aplicando en sus entradas las señales digitales de la figura A.11.

7.19. Obtener el cronograma de salida del multiplexor de 8 canales aplicando en sus entradas las señales digitales de la figura A.12.

7.20. Obtener el cronograma de salida del demultiplexor de 8 canales aplicando en sus entradas las señales digitales de la figura A.13.

7.21. Aplicar el cronograma de la figura A.11 a las entradas de un generador de paridad impar y obtener su salida gráficamente.

7.22. Diseñar un circuito generador de Hamming para 5 bits a transmitir.

8.1. Estudiar el funcionamiento del circuito de la figura A.14, para valores de x iguales a "0" y "1".

8.2. Estudiar el funcionamiento del circuito de la figura A.15, para valores de x iguales a "0" y "1".

8.3. Realizar la suma binaria de los números 1629,5 y 365,75.

8.4. Realizar la suma binaria de los números 33,17 y 417,62.

8.5. Realizar la resta binaria de los números 762,375 y 93,125.

8.6. Realizar la resta binaria de los números 62,375 y 193,768.

8.7. Calcular la representación en binario con complemento a 1 de los números decimales 132 y -132.

8.8. Convertir en binario con complemento a 1 el número decimal 0,818.

8.9. Calcular la representación en binario con complemento a 2 de los números decimales 132 y -132.

8.10. Convertir en binario con complemento a 2 el número decimal -0,818.

8.11. Realizar la resta binaria con complemento a 1 de los números 28,75 y 18,5.

8.12. Realizar la resta binaria con complemento a 1 de los números 80,125 y 83,25.

8.13. Realizar la resta binaria con complemento a 1 de los números 161,725 y 18,75.

8.14. Realizar la resta binaria con complemento a 1 de los números 18,75 y 161,725.

8.15. Realizar la resta binaria con complemento a 2 de los números 38 y 18.

8.16. Realizar la resta binaria con complemento a 2 de los números 281,5 y 338,5.

8.17. Realizar la resta binaria con complemento a 2 de los números 1614 y 18.

8.18. Realizar la resta binaria con complemento a 2 de los números 18,5 y 1614.

8.19. Realizar el problema 8.13 con bit de signo.

8.20. Realizar el problema 8.14 con bit de signo.

8.21. Realizar el problema 8.16 con bit de signo.

8.22. Realizar el problema 8.18 con bit de signo.

8.23. Realizar la resta en complemento a 2 de los números -40 y +49 disponiendo de registros de 6 bits.

8.24. Realizar la resta en complemento a 2 de los números -25 y +47 disponiendo de registros de 8 bits en los cuáles se incluye el bit de signo.

8.25. Realizar la resta en complemento a 2 con bit de signo de los números -40 y -49.

8.26. Realizar la resta en complemento a 2 con bit de signo de los números 25 y -47 utilizando registros de 6 bits.

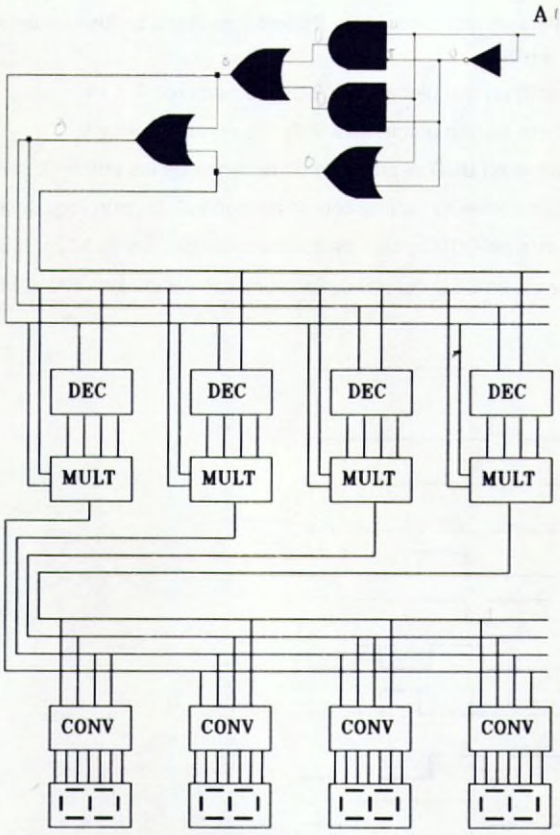


Figura A.9

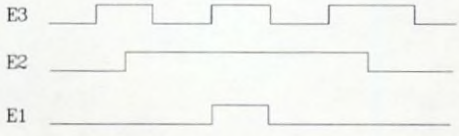


Figura A.10

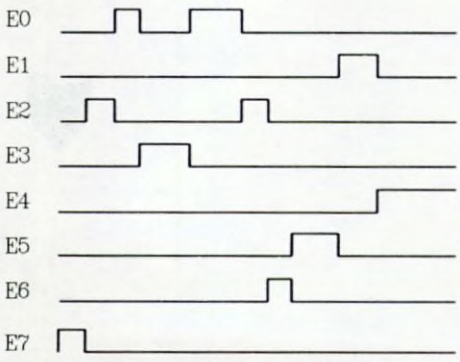


Figura A.11

8.27. Realizar la resta en complemento a 2 con bit de signo de los números 25 y -47 utilizando registros de 8 bits.

8.28. Realizar la resta en complemento a 2 de los números 0 y +47.

8.29. Realizar la resta en complemento a 2 de los números -40 y 0.

8.30. Obtener la suma en BCD natural con bit de signo de los números decimales 714 y 226.

8.31. Obtener la suma en BCD natural con bit de signo de los números decimales 1723 y 2460.

8.32. Obtener la resta en BCD natural de los números decimales 527 y -369.

8.33. Obtener la resta en BCD natural de los números decimales -128 y 345.

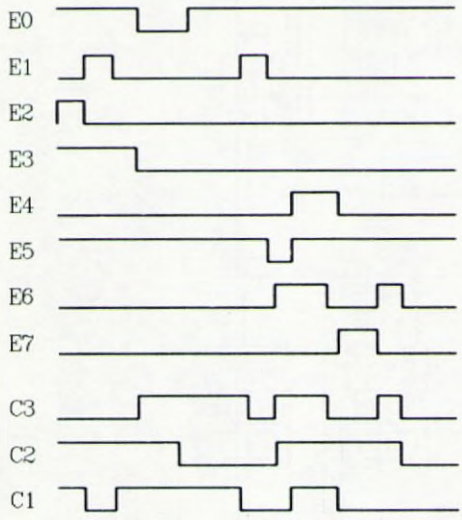


Figura A.12

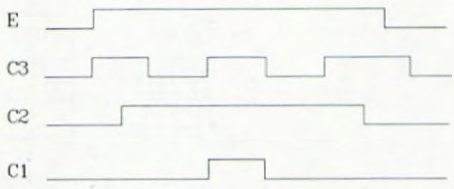


Figura A.13

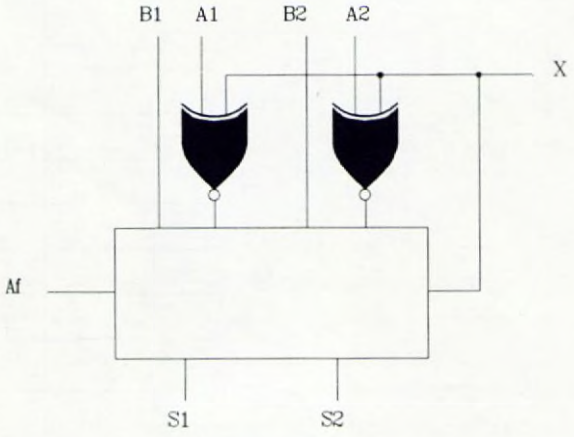


Figura A.14

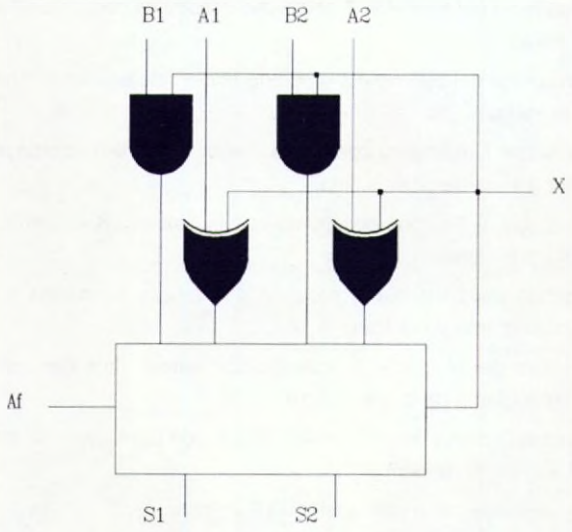


Figura A.15

8.34. Obtener la resta en BCD natural con bit de signo de los números decimales 732 y 460.

8.35. Obtener la resta en BCD natural con bit de signo de los números decimales 640 y 723.

8.36. Obtener la resta en BCD natural con bit de signo de los números decimales -16 y -23684.

8.37. Obtener la suma en BCD exceso 3 de los números decimales 3148 y 1715.

8.38. Obtener la suma en BCD exceso 3 con bit de signo de los números decimales 871 y 216.

8.39. Obtener la resta en BCD exceso 3 de los números decimales 2714 y 1866.

8.40. Obtener la resta en BCD exceso 3 con bit de signo de los números decimales 6820 y 4705.

8.41. Obtener la resta en BCD exceso 3 con bit de signo de los números decimales 280 y -5041.

8.42. Obtener la resta en BCD exceso 3 con bit de signo de los números decimales -17 y 13241.

8.43. Restar en BCD natural con bit de signo, los números octales 7251 y 402.

8.44. Sumar en BCD exceso 3 con bit de signo, los números hexadecimales 777 y FEA.

8.45. Obtener el resultado de la multiplicación de los números binarios 11001 y 1101.

8.46. Calcular la multiplicación de los números binarios 101011001 y 1001101 y expresar el resultado en sistema decimal.

8.47. Obtener el resultado de la división de los números binarios 1011101 y 1101.

8.48. Calcular la división de los números binarios 1101011011 y 111.

9.1. Analizar el esquema del biestable T asíncrono construido con un biestable J-K para cada valor de las entradas.

9.2. Analizar el esquema del biestable T síncrono construido con un biestable J-K para cada valor de las entradas.

9.3. Aplicar al biestable T asíncrono construido con biestable J-K el cronograma de la figura A.16 y obtener gráficamente su salida.

9.4. Aplicar al biestable T síncrono construido con biestable J-K el cronograma de la figura A.17 y obtener gráficamente su salida.

9.5. Obtener la salida del biestable D activado por flancos de subida si se le aplican las señales del cronograma de la figura A.18.

9.6. Obtener la salida del biestable D activado por flancos de bajada si se le aplican las señales del cronograma de la figura A.19.

9.7. Analizar el funcionamiento de un biestable D activado por flancos de subida si se conecta la entrada D a la señal de reloj.

9.8. Construir un biestable Latch con biestables T asíncronos.

9.9. Aplicar el cronograma de la figura A.20 a las entradas de un biestable S-R y obtener gráficamente su salida.

9.10. Construir un biestable D con biestables J-K.

9.11. Aplicar el cronograma de la figura A.21 a las entradas de un biestable Latch y obtener gráficamente su salida.

9.12. Analizar el funcionamiento de un biestable D activado por flancos de bajada si se conecta la entrada D a la señal de reloj.

9.13. Aplicar el cronograma de la figura A.22 a la entrada de un biestable T y obtener gráficamente su salida.

9.14. Aplicar el cronograma de la figura A.23 a las entradas de un biestable D activado por flancos de subida y obtener gráficamente su salida.

9.15. Aplicar el cronograma de la figura A.24 a las entradas de un biestable J-K y obtener gráficamente su salida.

9.16. Aplicar el cronograma de la figura A.25 a las entradas de un circuito de inscripción prioritaria y obtener gráficamente su salida.

9.17. Aplicar el cronograma de la figura A.26 a las entradas de un circuito de borrado prioritario y obtener gráficamente su salida.

9.18. Aplicar el cronograma de la figura A.27 a las entradas de un biestable D activado por flancos de bajada que ha sido cargado asíncronamente con un "1" y obtener gráficamente su salida.

9.19. Aplicar el cronograma de la figura A.28 a las entradas de un biestable J-K que ha sido cargado asíncronamente con un "1" y obtener gráficamente su salida.

9.20. El circuito de la figura A.29 tiene las entradas representadas por el cronograma de la figura A.30. Obtener gráficamente sus salidas.

9.21. Obtener las salidas S_3, S_2 y S_1 del circuito de la figura A.31 teniendo en cuenta que la entrada E toma los valores indicados en el cronograma de la figura A.32.

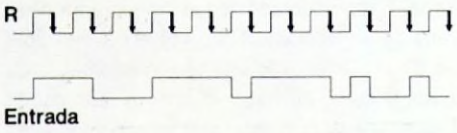


Figura A.16

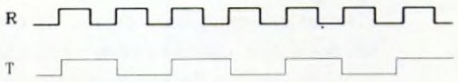


Figura A.17

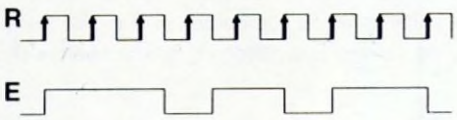


Figura A.18

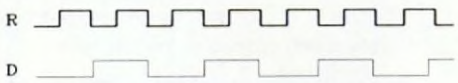


Figura A.19

Figura A.20

Figura A.21

Figura A.22

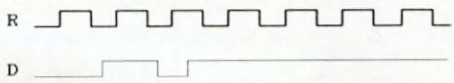


Figura A.23

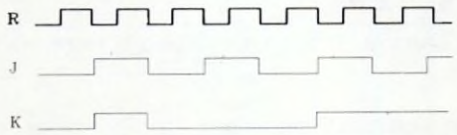


Figura A.24

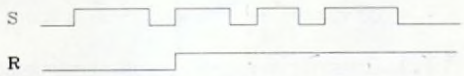


Figura A.25

Figura A.26

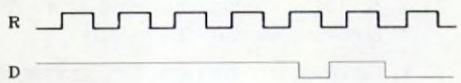


Figura A.27

9.22. Se dispone de una señal de reloj y de una señal E representadas en el cronograma de la figura A.33. Esta señal E se aplica a la entrada de un biestable T. La salida del biestable T se aplica simultáneamente a las entradas de un circuito de inscripción prioritaria. La salida de éste último se conecta con la entrada D de un biestable D sincronizado con la señal de reloj por flancos de bajada. Sus salidas (Q y su complemento) se conectan a las entradas D y C respectivamente de un biestable Latch. La salida del Latch se conecta a los terminales J y K de un biestable J-K sincronizado con la señal de reloj. Obtener gráficamente las salidas de todos los biestables.

10.1. Si se carga previamente un contador asíncrono con el valor decimal 5, y se desactiva el circuito un instante antes de que aparezca el flanco de bajada del tercer pulso de reloj, obtener el cronograma de funcionamiento y las señales que pueden quedar activadas en el circuito cuando ya no funcione.

10.2. Construir un contador síncrono binario de 8 bits.

10.3. Obtener el cronograma de salida de un contador de anillo de 4 etapas si todos los biestables se cargan previamente con un "1".

10.4. Obtener el cronograma de un contador conmutado de 4 bits si el primer biestable se carga previamente con un "1".

10.5. Obtener el cronograma de salida de un contador conmutado de 4 bits si todos los biestables se cargan previamente con un "1".

10.6. Obtener el cronograma de salida de un contador de anillo de 4 etapas si el primer biestable se carga previamente con un "1".

10.7. Obtener el cronograma de un contador síncrono binario de 3 bits cargado previamente con el valor binario "010" cuya entrada E siempre está a "1".

10.8. Obtener el cronograma de un contador síncrono binario de 3 bits cargado previamente con el valor binario "010" cuya entrada E siempre está a "0".

10.9. Obtener el cronograma de un contador BCD síncrono cargado previamente con el valor binario "0001" cuya entrada E siempre está a "1".

10.10. Obtener el cronograma de un contador BCD síncrono cargado previamente con el valor binario "0010" cuya entrada E siempre está a "0".

10.11. Obtener el cronograma de salida de un contador BCD Exceso-3 cuando se aplica en sus entradas las señales del cronograma de la figura A.34.

10.12. Diseñar un contador Up/Down binario con biestables D que funcione para la secuencia de valores del cronograma de la figura A.35.

11.1. Construir un registro de desplazamiento con biestables D síncronos.

11.2. Construir un registro de desplazamiento de 2 bits de salida y entrada serie con desplazamiento de derecha a izquierda con biestables J-K. La secuencia de entrada al registro será la combinación binaria "...00010110110000..." y cada señal dura un ciclo de reloj. Obtener gráficamente la salida.

11.3. Aplicar el cronograma de la figura A.36 a un registro de desplazamiento de 4 bits y obtener gráficamente su salida.

11.4. Construir un registro de desplazamiento de 8 bits.

11.5. Aplicar el cronograma de la figura A.37 a un registro de desplazamiento de 4 bits que ha sido previamente cargado con "1111".

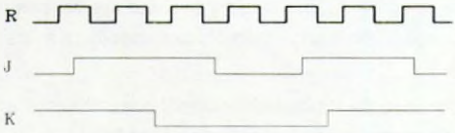


Figura A.28

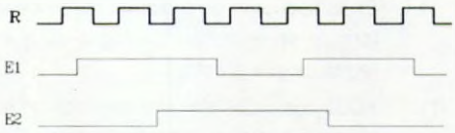


Figura A.30

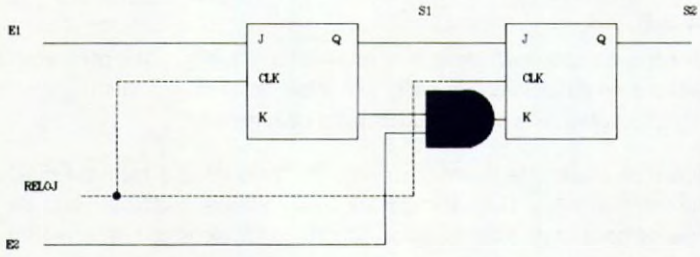


Figura A.29

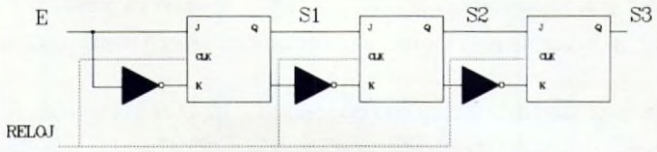


Figura A.31

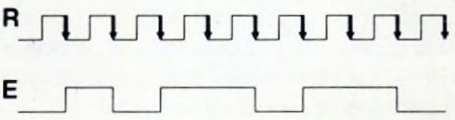


Figura A.32

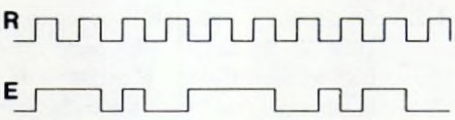


Figura A.33

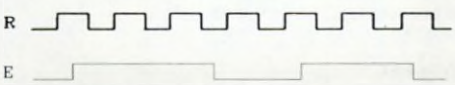


Figura A.34

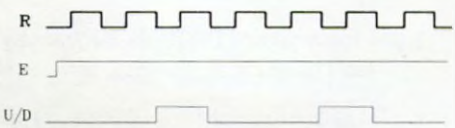


Figura A.35

11.6. En el esquema de registros conexionado de la figura A.38, indicar y explicar las distintas formas de conexión de registros que aparecen. En cada caso, hacer mención a los registros que afecta.

11.7. ¿Cuántos ciclos de reloj necesita un registro serie de 8 bits bidireccional para ponerse a cero mediante su entrada de datos si ha sido cargado previamente con "11111111"?.

11.8. ¿Cuántos ciclos de reloj necesita un registro paralelo de 8 bits para ponerse a cero mediante sus entradas de datos si ha sido cargado previamente con "11111111"?.

11.9. Realizar las conexiones necesarias para formar una estructura reticular con cinco registros, teniendo en cuenta que los registros R_1, R_2 y R_3 han de transmitir información a los registros R_4 y R_5.

11.10. Se dispone de una calle de 8 bits (8 líneas de 1 bit), y de seis registros de 8 bits conectados en paralelo con la calle. ¿La información de cuántos registros puede contener simultáneamente la calle en un instante determinado?.

12.1. Construir un circuito de direccionamiento de datos de una memoria que consta de los siguientes elementos: Decodificador de tres entradas, memoria RAM de 64 bits, con longitud de palabra de 8 bits y disposición 2D con núcleos de ferrita, registro de salida de datos de 8 bits, registro de entrada de datos de 8 bits.

12.2. Si en el problema anterior se desea grabar en la memoria los datos correspondientes a las sucesivas potencias de 2 (partiendo de n=0), indicar, analizar y explicar las señales necesarias para que aparezca en el registro de salida el valor "10000000". Realizar asimismo el proceso correspondiente a la lectura del dato ya grabado.

12.3. Realizar un esquema para formar una memoria de 1K con memorias elementales de 64 bits.

12.4. Realizar el problema 12.2 para una disposición 2 1/2 D de la memoria.

12.5. Seleccionar una línea ya grabada de una memoria ROM. Indicar gráficamente el proceso de selección y decodificación.

12.6. Realizar el problema 12.2 para una disposición 3D de la memoria.

12.7. Indicar gráficamente el proceso de grabación en una memoria RAM de 64 bits de la cifra binaria "01100111" en la posición 14 de la memoria.

13.1. Se dispone de una aplicación lógica programable FPLA de 3 entradas A, B, C y 3 salidas U, V, W. Se desea utilizar este circuito para generar las funciones que se detallan a continuación. Realizar el esquema de conexiones correspondientes a dichas funciones y la tabla que relaciona las entradas con las posiciones de memoria y las salidas.

$$U=ABC+\bar{A}BC+\bar{A}\bar{C}$$

$$V=ABC+\bar{A}\bar{B}$$

$$W=AB\bar{C}+A\bar{B}C$$

13.2. Programar una PROM de 3 entradas y 3 salidas y realizar el esquema de conexiones y la tabla de entradas y salidas, para las siguientes expresiones lógicas:

$$U=A\bar{B}C+\bar{A}B\bar{C}+ABC$$

$$V=ABC+A\bar{B}+B\bar{C}$$

13.3. Programar una RAM para grabar la información necesaria que sirva de activación posterior a un display siete segmentos conectado a su salida.

13.4. Programar una PROM con el mínimo gasto posible de elementos preprogramables de la memoria, empleando las ecuaciones del problema 13.1.

13.5. Programar una FPLA con el mínimo gasto posible de elementos preprogramables de la memoria, empleando las ecuaciones del problema 13.2.

13.6. Programar una PROM con el mínimo gasto posible de elementos preprogramables de la memoria, empleando las ecuaciones del problema 13.1.

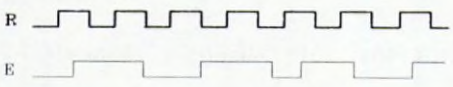


Figura A.36

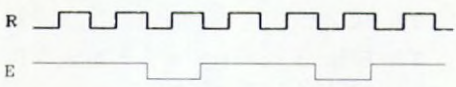


Figura A.37

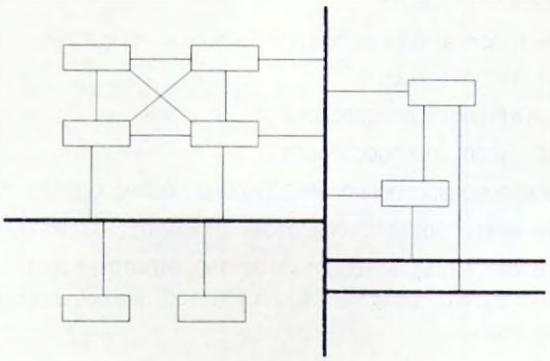


Figura A.38

SOLUCIONES A LOS PROBLEMAS

2.1. 1100110010

2.3. 1010,01100101

2.5. 51,71875

2.7. 173

2.9. 110010001001110

2.11. 3CB

2.13. 0,5217270243656

2.15. 1011111010111110

2.17. 52577401,732

2.19. 284

2.21. BCD-N: 0010001101110100 1000; BCD-A: 0010001111010100 1110; BCD-E:0101011010100 1111011;

Binario: 101110011000100; Hexadecimal: 5CC4

2.23. 62,5^{ns}

2.25. A los 48^{seg}

3.1. -12,625

3.3. Binario con signo y 5 bits C.F.: 011011001,11011; Binario y 10 bits C.F.: 11011001,1101100110;

Octal: 331,66314; Octal con signo: 0331,66314; Hexadecimal: D9,D99; Hexadecimal con signo: 0D9,D99

3.5. 1001010000011101

3.7. 011110000000100101100011

3.9. No se puede almacenar sin truncar parte de la cifra.

3.11. 101010010010110001100000

3.13. 010001001001011000110000000000000

3.15. 101010110101010001100000000000000

3.17. 11000000000000000000000000000000000000000000000000000000000000000000000000000000

3.19. 00001, 00010, 00100, 00111, 01000, 01011, 01101, 01110, 10000, 10011

3.21. 0000000, 1101001, 0101010, 1000011, 1001100, 0100101, 1100110, 0001111, 1110000, 0011001. No existe error en la transmisión 1001100, que corresponde al número 0100 en BCD (4 en decimal).

3.23. 11101011010

3.25. 7

3.27. 001

3.29. 01111

4.1. La ecuación final es:

$$\overline{A}C+\overline{A}\overline{D}$$

4.3. La ecuación final es:

$$\overline{A}+\overline{D}+\overline{B}\overline{C}$$

4.5. El proceso de simplificación es el siguiente:

$$B(\overline{A}(\overline{D}+D\overline{C})+(A+D\overline{A}C))=B(\overline{A}(\overline{D}+\overline{C})+A+DC)=B(\overline{D}+\overline{C}+A+DC)=B(\overline{D}+\overline{C}+A+C)=B$$

4.7. Siendo A y B las entradas de una puerta NOR, C y D las de la segunda puerta NOR y S la salida de la puerta NAND, se forma la siguiente tabla:

A	B	C	D	S
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

4.9. Siendo A y B las entradas de la puerta AND, S_1 la salida de la misma, conectada a los dos terminales de entrada NOR, y S_2 la salida de ésta última puerta:

A	B	S_1	S_2
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

4.11. Sean A, B y C las variables de entrada y S la salida:

A	B	C	S
0	0	0	0
0	0	0	1
0	0	1	1
0	0	1	1
0	1	0	1
0	1	0	1
0	1	1	1
0	1	1	1

4.13. S=1

4.15. Gráfico de la figura A.39.

4.17. Gráfico de la figura A.40.

4.19. Gráfico de la figura A.41.

4.21. S=0

4.23. S=0

4.25. Gráfico de la figura A.42.

5.1. Diagrama en figura A.43. La ecuación final es la siguiente:

$$F=\overline{A}+\overline{B}\overline{C}+\overline{B}\overline{D}+BCD$$

5.3. $F=m_0+m_1+m_3+m_6+m_7$; $F=M_2M_3M_5$

5.5. $F=m_0+m_1+m_2+m_3+m_4+m_5+m_6+m_7+m_8+m_9$; $F=M_0M_1M_2M_3M_4M_5$; la tabla de verdad es:

A	B	C	D	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

5.7. $F=m_1+m_4+m_5+m_8+m_9+m_{10}+m_{15}$; $F=M_1M_2M_3M_4M_8M_9M_{12}M_{13}M_{15}$

5.9. $F=M_2M_3M_4M_5M_6M_7M_{12}M_{13}M_{15}$

5.11. Tabla de verdad:

A	B	C	D	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

5.13. Ecuación final:

$$F=\bar{B}\bar{C}\bar{E}+\bar{A}B\bar{D}\bar{E}+\bar{A}\bar{B}CDE+\bar{A}B\bar{C}DE$$

5.15. Combinaciones de la tabla final: 0--1, -01-, que representan a todos los elementos y que conducen a la ecuación lógica:

$$F=(\bar{A}+D)(\bar{B}+C)$$

5.17. Mediante 4 negaciones se obtiene el gráfico de la figura A.44.

5.19. Mediante 2 negaciones se obtiene el gráfico de la figura A.45.

5.21. El diagrama coincide con la figura A.45.

Figura A.39

Figura A.40

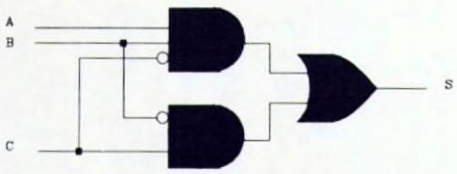


Figura A.41

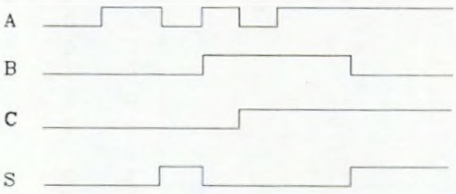


Figura A.42

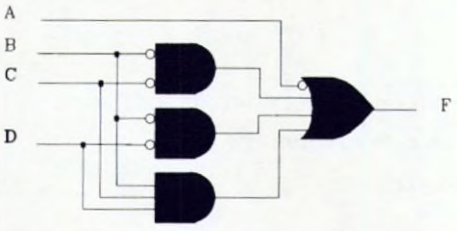


Figura A.43

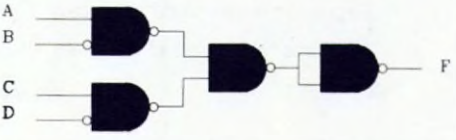


Figura A.44

5.23. Tabla de verdad:

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

5.25. La última tabla tiene los términos 000-, -001, 10-1, 1-11, que pueden simplificarse en la siguiente expresión algebraica:

$$F=(\bar{A}+\bar{B}+\bar{C})(A+\bar{B}+D)(A+C+D)$$

5.27. Ecuación final:

$$F=\bar{C}+D$$

Ver figura A.46

5.29. Ecuación obtenida:

$$F=A\bar{B}+\bar{B}C\bar{D}+\bar{A}C\bar{D}+ACD$$

5.31. Tabla de verdad:

A	B	C	D	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Expresión de suma de productos: $F=m_0+m_1+m_2+m_3+m_4+m_5+m_9+m_{11}+m_{12}+m_{13}$

La última tabla tiene el elemento -00- que conduce a la ecuación:

$$F=(\bar{B}+\bar{C})(\bar{A}+B+D)$$

5.33. La última tabla tiene el elemento 1-1- que conduce a la expresión:

$$F=AC+A\bar{C}\bar{D}+\bar{B}C\bar{D}+A\bar{B}D$$

Mediante 2 negaciones se obtiene el circuito de la figura A.47.

5.35. Ecuación que se obtiene:

$$F=(\bar{A}+\bar{B}+D)(\bar{A}+\bar{B}+C)(\bar{A}+B+\bar{C}+\bar{D})$$

5.37. Ecuación que se obtiene:

$$F=(\bar{A}+\bar{B}+D)(B+\bar{D}+\bar{E})(\bar{A}+\bar{C}+\bar{D}+\bar{E})(A+C+\bar{D}+\bar{E})$$

6.1. Figura A.48.

6.3. Figura A.49.

6.5. Figura A.50.

6.7. Figura A.51.

6.9. Figura A.52.

6.11. Figura A.53.

6.13. Figura A.54.

7.1. Según la disposición de selectores de la figura A.55, cualquier combinación aplicada simultáneamente al demultiplexor y al multiplexor seleccionarán idénticos terminales y la salida será siempre "1".

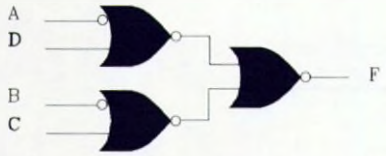


Figura A.45

Figura A.46

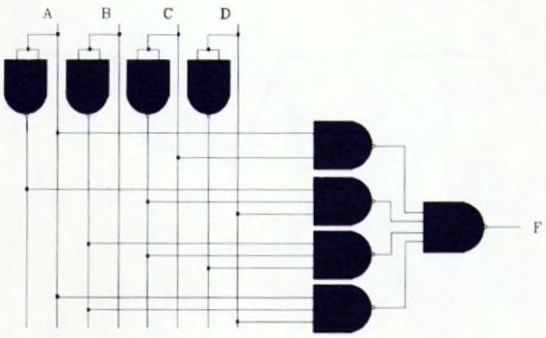


Figura A.47

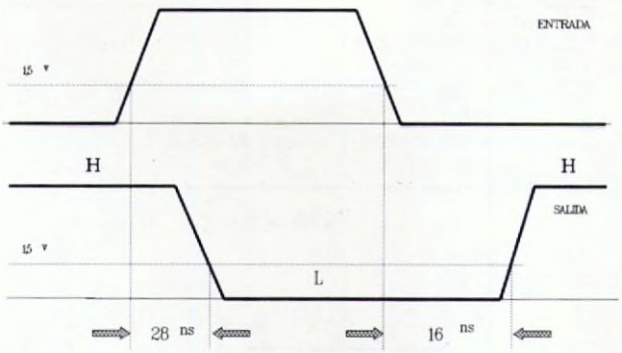


Figura A.48

7.3. Circuito de la figura A.56.

7.5. Circuito similar a la figura A.56 tomando de los decodificadores las salidas 0, 1, 2, 3, 6, 7, 9, 11 y 15.

7.7. Circuito similar a la figura A.56 pero con la estructura de 5 variables de la figura 7.6, tomando de los decodificadores las salidas 0, 2, 7, 8, 11, 12, 16 y 18.

7.9. Circuito de la figura A.57.

7.11. Circuito de la figura A.58.

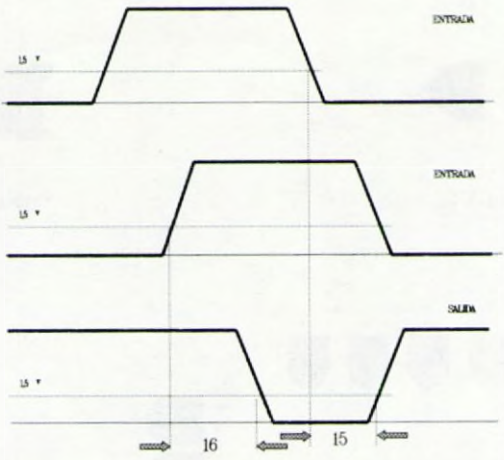


Figura A.49

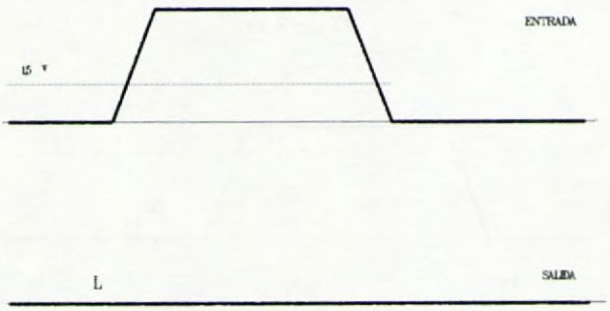


Figura A.50

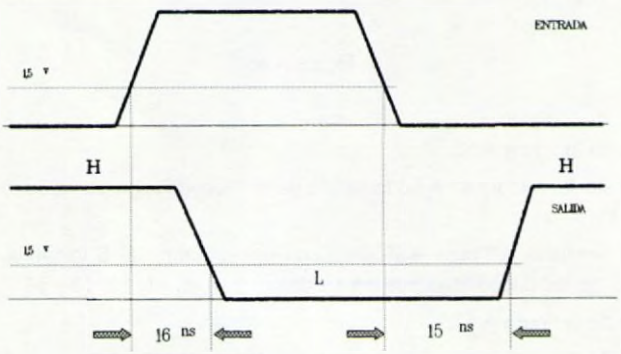


Figura A.51

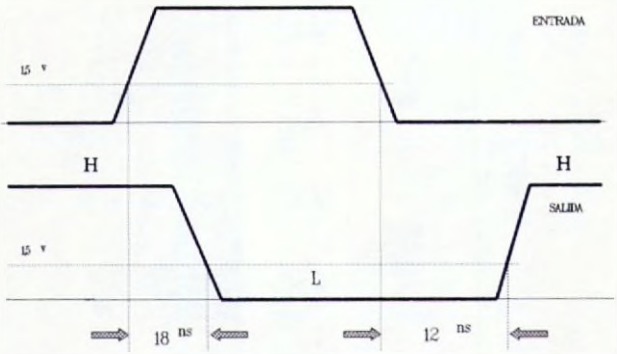


Figura A.52

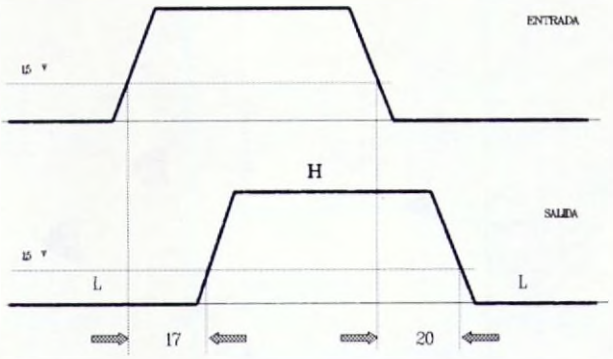


Figura A.53

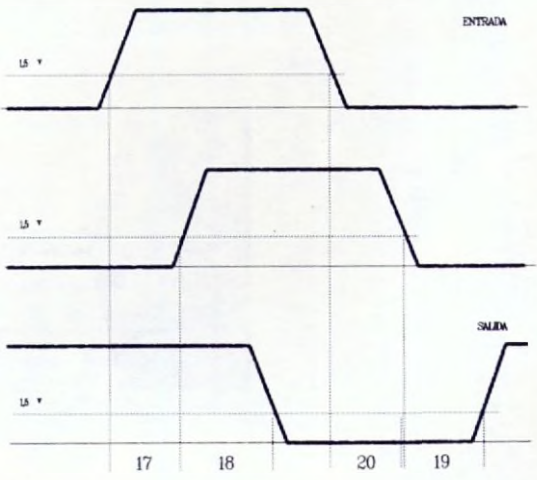


Figura A.54

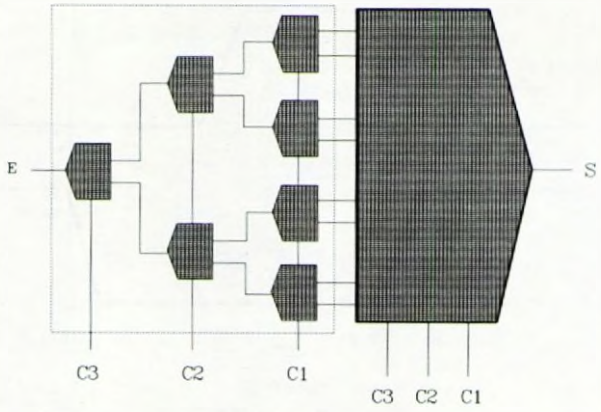


Figura A.55

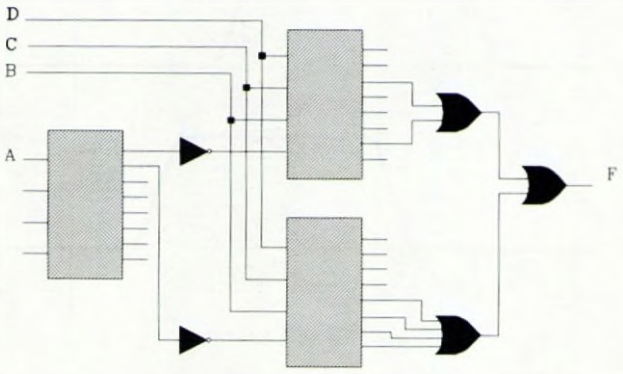


Figura A.56

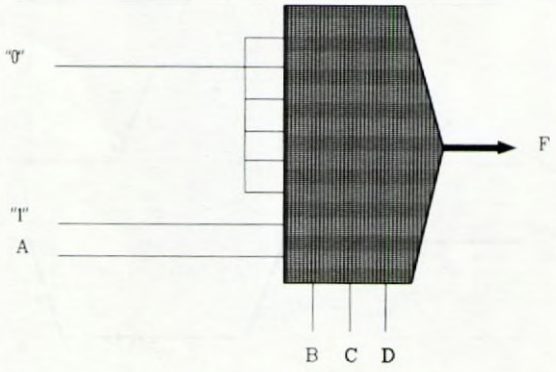


Figura A.57

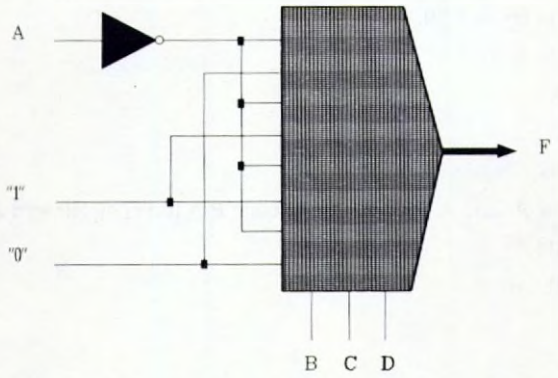


Figura A.58

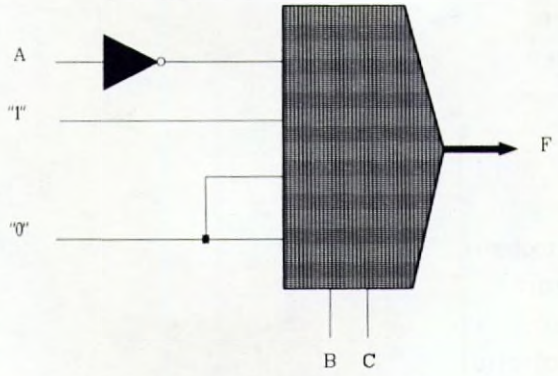


Figura A.59

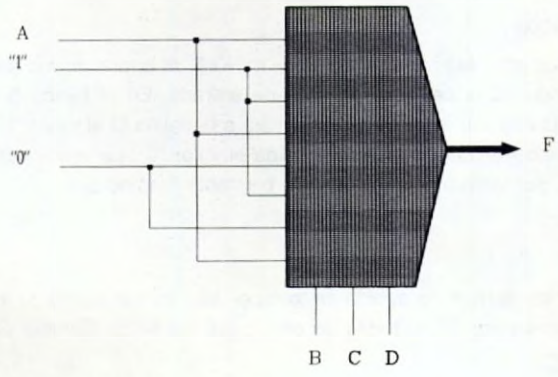


Figura A.60

7.13. Circuito de la figura A.59.

7.15. Circuito de la figura A.60.

7.17. Figura A.61.

7.19. Figura A.62.

7.21. Circuito A.63.

8.1. Para x=0, se obtiene A. Para x=1 se obtiene B-A (En complemento a 2)

8.3. 11111001011,01

8.5. 1010011101,010

8.7. 01111011

8.9. 01111100

8.11. 01010,01

8.13. 10001111,00

8.15. 010100

8.17. 11000111100

8.19. 010001111,00

8.21. 1000111001,0

8.23. Desbordamiento

8.25. 0001001

8.27. 01001000

8.29. -101000

8.31. 00100000110000011

8.33. -010001110011

8.35. 1000010000011

8.37. 0111101110010110

8.39. 0011101101111011

8.41. 110000110010101000

8.43. 000110100100101010

8.45. 101000101

8.47. 111,001001000

9.1. Para Q=0 las entradas toman valores J=1 y K=0. Al aparecer una señal en la entrada de reloj se producirá la captura de datos de entrada, En el flanco de bajada de la señal aplicada a la entrada de reloj se transmitirá a la salida Q el valor "1". Para Q=1 el efecto es el contrario transmitiéndose a la salida el valor "0". La salida cambia con cada pulso de entrada, por tanto funciona como un biestable T asíncrono.

9.3. Figura A.64.

9.5. Figura A.65.

9.7. Al coincidir los flancos de subida se toma el valor inmediatamente anterior en la entrada D, siendo este valor "0" en todas las ocasiones. La salida siempre será "0".

9.9. Figura A.66.

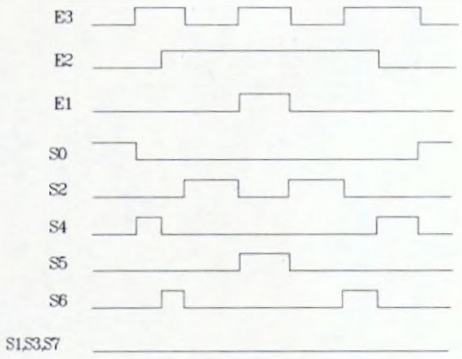


Figura A.61

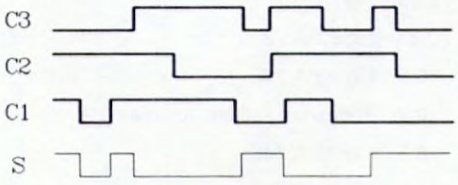


Figura A.62

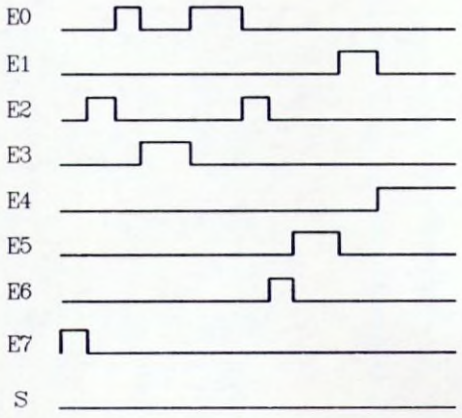


Figura A.63

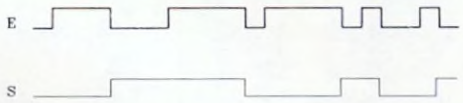


Figura A.64

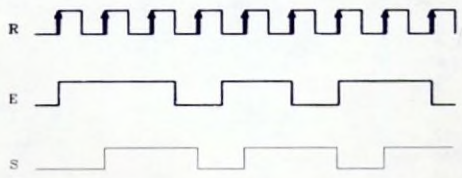


Figura A.65

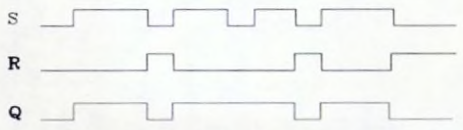


Figura A.66

9.11. Figura A.67.

9.13. Figura A.68.

9.15. Figura A.69.

9.17. Figura A.70.

9.19. Figura A.71.

9.21. Figura A.72.

10.1. Figura A.73.

10.3. Todas las salidas son siempre "1".

10.5. Figura A.74.

10.7. Figura A.75.

10.9. Figura A.76.

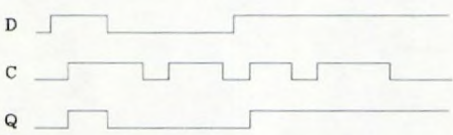


Figura A.67

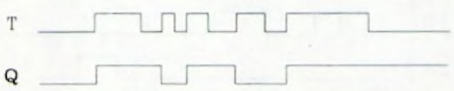


Figura A.68

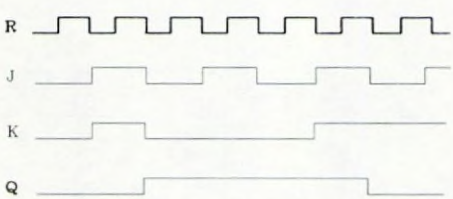


Figura A.69

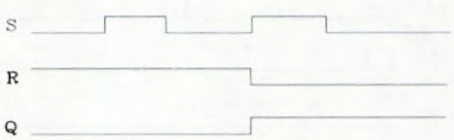


Figura A.70

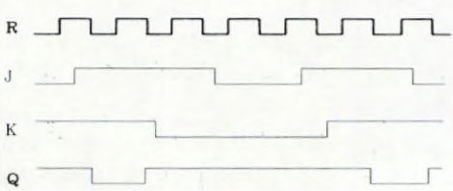


Figura A.71

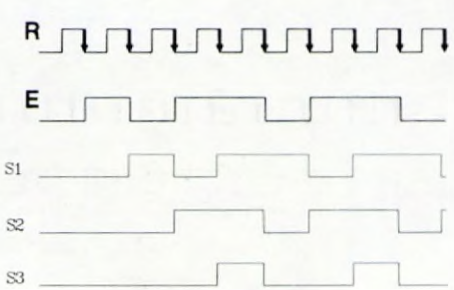


Figura A.72

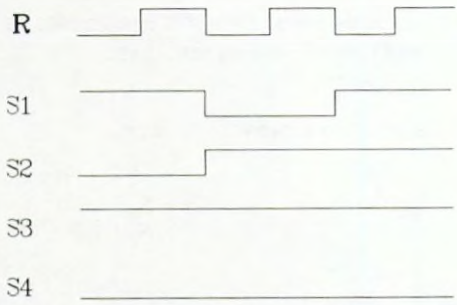


Figura A.73

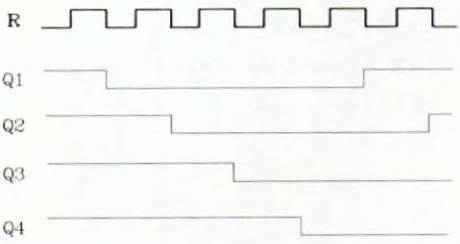


Figura A.74

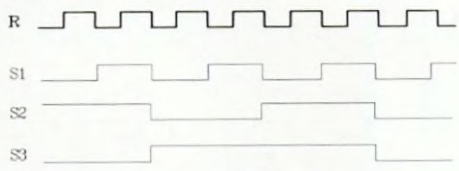


Figura A.75

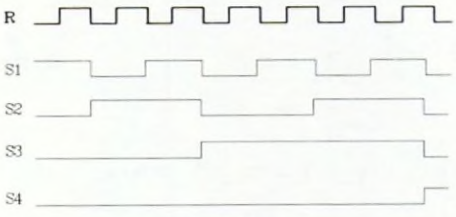


Figura A.76

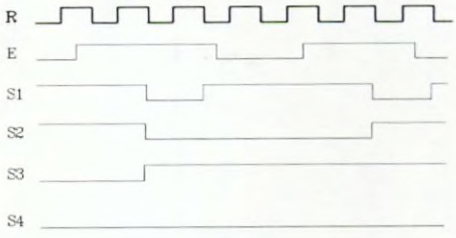


Figura A.77

10.11. Figura A.77.

11.1. Figura A.78.

11.3. Figura A.79.

11.5. Figura A.80.

11.7. Figura A.81.

11.9. 8

12.1. Figura A.82.

12.3. Es equivalente a la figura 12.18 utilizando 16 bloques de 64 bits, direccionados con un registro RD de 6 bits, lo que implica la utilización de dos decodificadores 3x8.

12.5. El decodificador de líneas selecciona la línea horizontal de la memoria y se envía al registro de salida auxiliar de la memoria. Si se emplean biestables D según la figura 12.16, se activa la señal de selección del biestable que junto con el valor de Q se envía al registro de salida.

12.7. Una memoria de 64 bits que pueda grabar palabras de 8 bits utiliza sólo 8 posiciones de memoria por lo que no existe la línea o posición 14 en este tipo de estructura.

13.1. Figura A.83.

13.3. Igual que las ROM y PROM del tema 13. Es la tabla del convertidor de código.

13.5. Figura A.84.

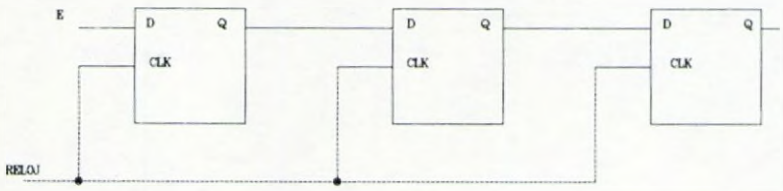


Figura A.78

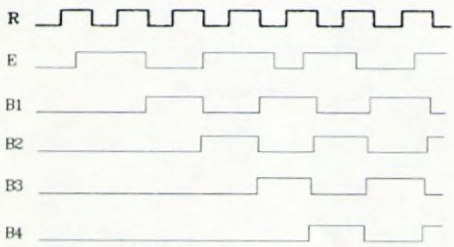


Figura A.79

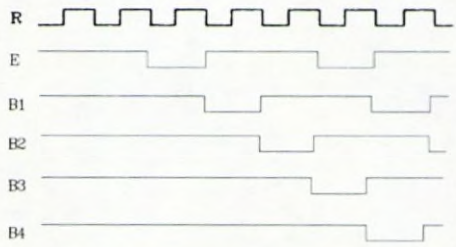


Figura A.80

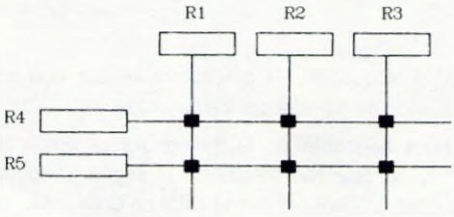


Figura A.81

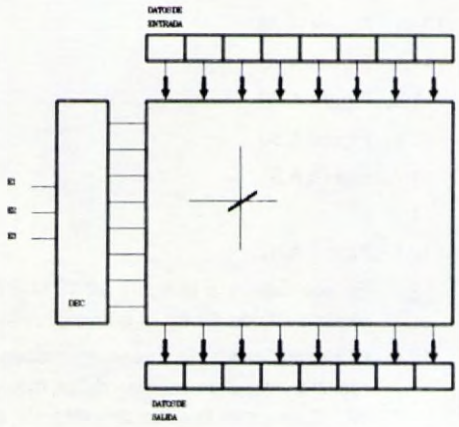


Figura A.82

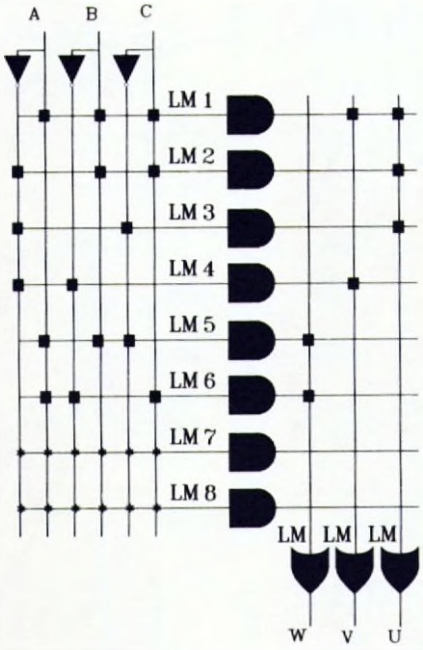


Figura A.83

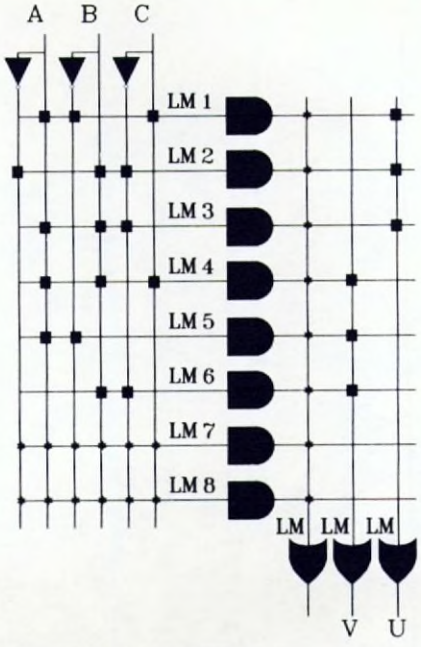


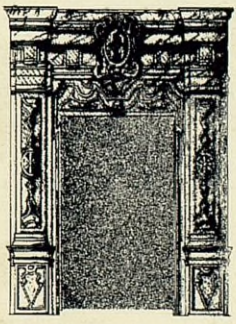
Figura A.84

APENDICE B
BIBLIOGRAFIA

APENDICE B

BIBLIOGRAFIA

- D.L. CANNON, *A fondo: Electrónica del estado sólido II* /Ed. Anaya
- R. WALKER, *Aplicaciones de los ordenadores* /Ed. Anaya
- H. TAUB, *Circuitos digitales y microprocesadores* /Ed. Mc Graw Hill
- J. HOLGADO, *Circuitos y Sistemas digitales (Vol 1 a 4)* /S. Public. E.U. Politécnica Cádiz
- W. KLEITZ, *Digital and microprocessor fundamentals* /Ed. Prentice Hall
- DESCHAMPS-ANGULO, *Diseño de sistemas digitales* /Ed. Paraninfo
- G. WOLF, *Electrónica digital* /Ed. Marcombo
- J.M. ANGULO, *Electrónica digital moderna* /Ed. Paraninfo
- F. ALDANA - R. ESPARZA - P.M. MARTINEZ, *Electrónica Industrial: Técnicas Digitales* /Ed. Marcombo
- MEINADIER, *Estructura y funcionamiento de los computadores digitales* /Ed. AC
- P. de MIGUEL ANASAGASTI, *Fundamentos de los computadores* /Ed. Paraninfo
- R.L. TOKHEIM, *Fundamentos de los microprocesadores* /Ed. McGraw Hill
- E. MANDADO, *Manual de prácticas de electrónica digital* /Ed. Marcombo
- R. ZAKS, *Microprocessor interfacing techniques* /Ed. Sibex
- R.L. TOKHEIM, *Principios digitales* /Ed. McGraw Hill
- MALVINO, *Principios y aplicaciones digitales* /Ed. Marcombo
- R. ZAKS, *Programación del Z-80* /Ed. Anaya
- E. MANDADO, *Sistemas electrónicos digitales* /Ed. Marcombo
- *TTL Data Book*, Texas Instrument



SERVICIO DE PUBLICACIONES
UNIVERSIDAD DE CADIZ